Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan tutkinto-ohjelma

Rami Aamulehto

Digitaalisten aikakaus- ja sanomalehtien julkaiseminen HTML5-tekniikalla

Diplomityö
Espoo, 1. maaliskuuta 2013

Valvoja: Professori Pirkko Oittinen
Ohjaaja: Diplomi-insinööri Mikko Kuhna

Aalto University
School of Science
Degree Programme of Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author: Rami Aamulehto

Title:
Digital Magazine and Newspaper Publishing with HTML5

Date: March 1, 2013 **Pages:** xiii + 145

Professorship: Media Technology **Code:** T-75

Supervisor: Professor Pirkko Oittinen

Instructor: Mikko Kuhna, M.Sc. (Tech.)

The aim of this thesis has been to study the use of HTML5 technology in digital magazine and newspaper publishing. The field of research has been focused primarily on tablet and mobile platforms which have played a crucial role in the convergence of devices, media and technologies during the past years. This development has increased the significance of browser based applications, contents and services which can flow across different devices and be reached regardless of time, place and device.

While browsers have been adopting the new HTML5 features, new execution possibilities have been emerged in digital publishing for tablet and mobile platforms. These implementations can match the performance of native applications and at the same time streamline the publishing process. The lack of de facto implementation methods raises a need for comparison which has been addressed in this study.

The most important technologies and publishing methods for HTML5 based content have been identified and introduced in this thesis. The empirical research was based on the implementation of Aalto University Magazine tablet concept: the visual design of the concept magazine was provided by Juho Hiilivirta. The HTML5 based magazine was used as a part of the implemented iOS and web applications which served as research platforms for different implementation methods.

Performance and feature detection tests were designed and executed using a range of browsers and devices from desktop, tablet and mobile platforms. The results of this study include Stage Framework for publishing HTML5 based content and recommended conventions for digital publishing in addition to the implemented applications, demo magazine and test results. All in all the new browser technologies were well supported across different platforms offering an excellent publishing platform as the most common execution environment in the world. In the light of this study publishers and developers are encouraged for a confident adoption of these technologies.

Keywords: digital publishing, magazine, newspaper, tablet, mobile, HTML5, CSS3, web application

Language: Finnish

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan tutkinto-ohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Rami Aamulehto		
Työn nimi:	Digitaalisten aikakaus- ja sanomalehtien julkaiseminen HTML5-tekniikalla		
Päiväys:	1. maaliskuuta 2013	**Sivumäärä:**	xiii + 145
Professuuri:	Mediatekniikka	**Koodi:**	T-75
Valvoja:	Professori Pirkko Oittinen		
Ohjaaja:	Diplomi-insinööri Mikko Kuhna		

Tämän diplomityön tavoitteena on ollut tutkia HTML5-tekniikan käyttöä digitaalisten aikakaus- ja sanomalehtien julkaisemisessa. Tutkimusalueena ovat olleet ensisijaisesti tabletti- ja mobiilialustat, jotka ovat olleet tärkeässä roolissa viime vuosina tapahtuneessa asiakaslaitteiden, medioiden ja teknologioiden konvergenssissa. Tämä kehitys on kasvattanut selainpohjaisten sovellusten, sisältöjen ja palveluiden merkitystä, jotka voivat kulkea eri laitteiden välillä ja olla käyttäjien saavutettavissa ajasta, paikasta ja laitteesta riippumatta.

Selainten kehittyessä ja adoptoidessa HTML5:n uusia ominaisuuksia, on digitaalisten lehtien julkaisemisessa tabletti- ja mobiilialustoille avautunut uusia toteutustapoja, jotka voivat vastata natiivien sovellusten suorituskykyyn ja samalla helpottaa itse julkaisuprosessia. De facto -toteutuksen puuttuessa herää tarve eri toteutustapojen vertailulle, johon tässä diplomityössä on pyritty vastaamaan.

Diplomityössä on esitelty keskeiset teknologiat ja toteutusratkaisut HTML5-taitetun sisällön julkaisemiselle. Julkaisuformaatin kokeellista tutkimusta varten toteutettiin Aalto University Magazine -lehdelle tablettikonsepti Juho Hiilivirran suunnitteleman visuaalisen ilmeen pohjalta. HTML5-taitettua demolehteä hyödynnettiin toteutettujen iOS-sovelluksen ja web-applikaation osana, jotka palvelivat alustoina eri toteutusratkaisujen tutkimiselle.

Diplomityössä tutkittiin eri selainteknologioiden soveltuvuutta ja suorituskykyä osana alusta- ja suorituskykytestausta, joka suoritettiin laajalle selainkirjolle työpöytä-, tabletti- ja mobiilialustoilla. Diplomityön keskeisiin tuloksiin kuuluvat toteutettujen sovellusten, demolehden ja testien tulosten lisäksi Stage Framework -ohjelmistokehys HTML5-taitettujen lehtien julkaisemiselle sekä suositeltujen käytäntöjen lista digitaaliselle julkaisemiselle. Kaiken kaikkiaan uudet selainteknologiat olivat hyvin tuettuja alustasta riippumatta tarjoten erinomaisen julkaisualustan maailman levinneimpänä suoritusympäristönä. Tämän diplomityön tulosten valossa julkaisijoita ja kehittäjiä kannustetaan näiden teknologioiden rohkeaan adoptointiin.

Asiasanat:	digitaalinen julkaiseminen, aikakauslehti, sanomalehti, tabletti, mobiili, HTML5, CSS3, web-applikaatio
Kieli:	Suomi

Alkusanat

Tämä diplomityö on kirjoitettu Aalto-yliopiston perustieteiden korkeakoulun mediatekniikan laitoksella osana Next Media -projektia. Suuret kiitokset työn mahdollistamisesta kuuluvat visuaalisen median tutkimusryhmän professori Pirkko Oittiselle ja tohtorikoulutettava Mikko Kuhnalle. Kiitos ainutlaatuisesta tilaisuudesta ja erinomaisesta ohjauksesta.

Kiitos myös perheelleni tuesta ja ohjauksesta, ystävilleni hauskoista hetkistä ja inspiraatiosta. Erityiskiitos upealle vaimolleni ymmärryksestä, rakkaudesta ja maailman parhaista ideoista tilanteessa kuin tilanteessa. Kiitän teitä toivottavasti muussa yhteydessä paremmin kuin T-talon kirjastosta löytyvän mustan kirjan sivuilla.

Espoo, 1. maaliskuuta 2013

Rami Aamulehto

Lyhenteitä ja käsitteitä

AAC	Advanced Audio Coding, häviöllinen pakkausstandardi digitaaliselle äänelle, jota muun muassa Chrome, Safari ja Internet Explorerin uusimmat versiot tukevat HTML5:n audio-elementissä
AUM	Aalto University Magazine, Aalto-yliopiston viestinnän julkaisema sidosryhmälehti
CSS	Cascading Style Sheets, tyylikuvauskieli HTML-dokumenttien muotoilemiseen
CSS Hyphenation	CSS-spesifikaation mukainen tekstin automaattinen tavutus
CSS3	W3C:n julkaiseman ja ylläpitämän CSS-spesifikaation kolmas sukupolvi, määrittelee selainten noudattaman CSS-tyylikuvauskielen
CSS3 3D Transform	Oman spesifikaation määrittelemä 3D-transformaatio (3D-muuntaminen) DOM-elementtien ulkoasun manipulointiin, esimerkiksi valinnaisen akselin ympäri kiertämiseen
CSS3 Animation	Oman spesifikaation määrittelemä animointi DOM-elementtien ulkoasun manipulointiin, esimerkiksi läpinäkyvyyden temporaalinen muuttaminen
CSS3 Media Queries	Oman spesifikaation määrittelemä mediakysely, jolla voidaan ohjelmallisesti tiedustella asiakaslaitteen ominaisuuksia, kuten orientaatiota

CSS3 Transform	Oman spesifikaation määrittelemä 2D-transformaatio (2D-muuntaminen) DOM-elementtien manipulointiin ilman syvyysakselia
CSS3 Transition	Oman spesifikaation määrittelemä siirtymä-tekniikka, jolla voidaan animoida DOM-elementin haluttuja ominaisuuksia, kuten esimerkiksi leveyttä asettamalla haluttu siirtymäaika ja antamalla uusi leveyden arvo
DOM	Document Object Model, tekniikka HTML-sivun sisällön esittämiseen ja muokkaamiseen ohjelmallisesti, oliopohjaisesti, sekä XML-hierarkiaa tukien
DOM Touch Events	DOM-kosketustapahtumat, selaimen tukema tekniikka, jolla asiakaslaitteen kosketukset voidaan tunnistaa ja ohjata kosketuskohdan mukaan oikeille DOM-elementeille
EOT	Embedded OpenType, Microsoftin julkaisema kevyempi versio OpenType -kirjasintyypistä käytettäväksi verkkosivuilla, jota tuetaan pääasiassa Microsoftin omissa selaimissa
GIF	Graphic Interchange Format, vanha 256:een väriin rajoittuva tiedostomuoto kuville
H.264	H.264/MPEG-4 AVC, de facto -standardi videoiden pakkaamiseen verkossa - tukee enimmillään 4K-resoluutiota (4096x2304)
H.265	High Efficiency Video Coding, HEVC, työstettävänä oleva standardi videoiden pakkaamiseen, joka tulee aikanaan korvaamaan H.264:n eli AVC-standardin - tukee enimmillään 8K-resoluutiota (8192x4320)
JPEG	Joint Photographic Experts Group, häviöllistä pakkausta käyttävä tiedostomuoto kuville
JS	JavaScript, web-selaimessa käytettävä komentosarjakieli, jolla voidaan muun muassa muokata HTML-sivun DOM-elementtejä

JSON	JavaScript Object Notation, JavaScript-objektinotaatio, tiedonsiirtoon käytettävä ihmisluettava tekstimuoto
HTML	Hypertext Markup Language, verkkosivujen kuvauskieli, joka noudattaa W3C:n julkaisemaa ja ylläpitämää HTML-spesifikaatiota
HTML5	W3C:n julkaiseman ja ylläpitämän HTML-spesifikaation viides versio, joka päivitti edeltävää useilla täysin uusilla ominaisuuksilla, kuten videota ja ääntä tukevilla elementeillä, sekä animaatioita tukevalla piirtoalueella. Myös käsite laajemmalle teknologiselle kontekstille
HTML5 Application Cache	HTML5-spesifikaation määrittelemä sovellusvälimuisti verkkosivujen offline-käyttöä varten, tallennettavat resurssit määritellään verkkosivukohtaisella manifest-tiedostolla
HTML5 Audio Element	HTML5-spesifikaation määrittelemä audio-elementti, joka tukee selaimesta riippuen muun muassa seuraavia audio-formaatteja: MP3, AAC, WAV PCM, WebM Vorbis, Ogg Vorbis, Ogg Opus. Elementti mahdollistaa äänen toistamisen selaimen tukemilla formaateilla
HTML5 Canvas Element	HTML5-spesifikaation määrittelemä piirtoalue-elementti, jonka sisältö esitetään selaimessa rasterigrafiikkana
HTML5 Video Element	HTML5-spesifikaation määrittelemä video-elementti, joka tukee selaimesta riippuen muun muassa seuraavia video-formaatteja: Ogg Theora, H.264/MPEG-4 AVC, VP8/WebM. Elementti mahdollistaa videon toistamisen selaimen tukemilla formaateilla
MP3	Moving Picture Experts Groupin (MPEG) suunnittelema häviöllinen pakkausstandardi digitaaliselle äänelle
MPEG-4	Moving Picture Experts Groupin (MPEG) julkaisema standardi audiovisuaalisen datan pakkaamiselle

Ogg	Säiliötiedostomuoto audiovisuaaliselle sisällölle, joka tukee useita pakkausformaatteja. Audiosisältö voidaan tallentaa häviöllisillä Speedx, Vorbis tai Opus -kodekeilla, häviötömällä FLAC-kodekilla tai pakkaamattomana OggPCM:llä. Videosisältö voidaan puolestaan tallentaa häviöllisillä Theora, Darkin tai Dirac -kodekeilla, häviöttömällä Diracilla tai pakkaamattomana OggUVS:llä
OTF	OpenType, Microsoftin kehittämä skaalautuva kirjasinformaatti, joka on TTF:stä kehitetty paranneltu versio
PNG	Portable Network Graphics, häviötön tiedostomuoto kuville, joka tukee läpinäkyvyyttä
SDK	Software Development Kit, ohjelmistokehityspaketti
SVG	Scalable Vector Graphics, XML-kieleen perustuva vektorikuvien kuvausformaatti, joka tukee myös animaatioita
TTF	TrueType, Applen kehittämä kirjasinformaatti, joka on ollut hyvin yleinen Apple (Mac) OS X ja Microsoft Windows -käyttöjärjestelmissä
VP8	Googlen omistama pakkausformaatti digitaaliselle videolle, jota osa selaimista tukee WebM-tiedostosäiliöllä HTML5:n video-elementissä
W3C	World Wide Web Consortium, verkon standardeja, kuten HTML ja CSS julkaiseva ja ylläpitävä konsortio
WAV PCM	Wave Form Audio File Format, tiedostomuoto äänen tallentamiseen, Pulse Code Modulation eli pulssikoodimodulaatio viittaa käytettyyn äänen digitalisointimenetelmään, jossa analogisesta signaalista otetaan näytteitä tasaisin väliajoin

Web Fonts Web-kirjasimet viittaavat verkossa käytettäviin kirjasinformaatteihin, jotka mahdollistettiin CSS-spesifikaation toisessa sukupolvessa. Kirjasimia varten selaimeen ladataan tietyn kirjasinformaatin tiedosto, joka kuvailee käytettävän kirjasimen merkit

Web Storage HTML5-spesifikaatiosta omaksi määrittelyksi siirretty asiakaspuolen (selaimen) tallennustila verkkosivuille, josta selaimet implementoivat yleisesti Local Storage ja Session Storage -tallennustilat: Session Storagen soveltuessa tietojen tallentamiseen yhden vierailun ajaksi ja Local Storagen tallentaessa tietoa domain-kohtaisesti kunnes tiedot poistetaan käyttäjän tai verkkosivun toimesta

Web Workers Taustasuoritustekniikka JavaScript-komentosarjojen suorittamiseksi taustalla, erillään käyttöliittymää ohjaavasta JavaScriptistä

WebGL API WebGL-rajapinta (Web Graphics Library), selaimien tukema JavaScript-rajapinta interaktiiviselle kaksi- ja kolmiulotteiselle grafiikalle, joka perustuu OpenGL ES 2.0 -rajapinnalle, joka on osa näytönohjainten tukemaa OpenGL-rajapintaa

WebM Googlen omistama tiedostosäiliöformaatti, jonka kanssa käytetään yleisesti VP8-pakkausta videolle ja Vorbis-pakkausta äänelle

WOFF Web Open Font Format, Mozilla Foundationin, Opera Softwaren ja Microsoftin kehittämä verkossa käytettävä kirjasinformaatti, josta odotetaan verkon hallitsevaa kirjasinformaattia. Formaatista on tulossa W3C:n suositus käytettäväksi kirjasinstandardiksi verkossa

XML Extensible Markup Language, datan esitysmuodon määrittelevä formaatti, jossa data esitetään metadatan sisältävien tagien sisällä

Sisältö

Luku 1

Johdanto

Tässä diplomityössä käsitellään digitaalisten aikakaus- ja sanomalehtien julkaisemista HTML5-tekniikalla. Mobiiliselainten kehittyessä ja adoptoidessa HTML5:n uusia ominaisuuksia, on digitaalisten lehtien julkaisemisessa tabletti- ja mobiilialustoille avautunut uusia toteutustapoja, jotka voivat vastata natiivien sovellusten suorituskykyyn ja samalla helpottaa itse julkaisuprosessia. Tämä konvergenssi tekee HTML5-taitetun sisällön jakelusta yhä houkuttelevamman vaihtoehdon. De facto -toteutuksen puuttuessa herää tarve eri toteutustapojen vertailulle, johon tässä diplomityössä pyritään vastaamaan.

Diplomityön keskeisiin tuloksiin kuuluvat optimaalisten julkaisukäytäntöjen kartoittaminen, ohjelmistokehys HTML5-taitettujen aikakaus- ja sanomalehtien julkaisemiselle, sekä alusta- ja suorituskykytestauksen tulokset tärkeimmillä tabletti- ja mobiilialustoilla. Diplomityön tuloksena syntynyt Stage Framework -ohjelmistokehys on saatavilla avoimilla lisensseillä ja on siten kenen tahansa käytettävissä.

Tämä diplomityö tuloksineen on toteutettu silmällä pitäen tapausesimerkkiä, jossa lehtiä julkaiseva taho haluaisi julkaista lehtensä digitaalisesti mobiilialustoille ilman, että joutuisi implementoimaan sovelluksen jokaiselle alustalle erikseen. HTML5 näyttäisi tarjoavan alustariippumattoman ratkaisun, mutta kuinka sovellus kannattaisi toteuttaa? Mitä seikkoja toteutuksessa tulisi ottaa huomioon?

Tässä luvussa esitellään jäljempänä tutkimuskysymykset, käytetty aineisto mukaan lukien tulokset, sekä tutkimuksen taustaa. Diplomityön toisessa luvussa paneudutaan digitaaliseen julkaisemiseen ja sen tarjoamiin vaihtoehtoihin. Kolmannessa luvussa tutustutaan käytettävissä oleviin ja diplomityössä käytettyihin toteutustekniikoihin ja teknologioihin. Neljännessä luvus-

sa käydään lävitse keskeisimmät referenssijulkaisut tabletti- ja mobiilialustoilla, ja tutustutaan niissä valittuihin ratkaisuihin. Viidennessä luvussa esitellään diplomityössä toteutetut implementaatiot esimerkkinä käytetyn demolehden sisällön julkaisemiseksi, kun taas kuudennessa luvussa käsitellään tuloksia, erityisesti toteutettua alusta- ja suorituskykytestausta. Diplomityön viimeiset luvut koostuvat analyysista ja tulkinnasta, sekä kaiken yhteen sitovasta lopuksi-luvusta.

1.1 Tutkimuskysymykset

Diplomityön aluksi määriteltiin kolme tutkimuskysymystä, joilla pyrittiin haarukoimaan HTML5-taitettujen digitaalisten lehtien julkaisemiseen liittyvää ongelmakenttää. Tutkimuskysymykset eivät perinteisessä mielessä määrittäneet diplomityön tuloksia, sillä esimerkiksi toteutettavat sovellukset oli päätetty ennalta. Tutkimuskysymykset kuitenkin ohjasivat tutkimuksellista näkökulmaa ja konkretisoivat HTML5-julkaisemiseen liittyvät ongelmat. Tutkimuskysymykset olivat seuraavat:

- *Mikä on optimaalisin keino tarjoilla HTML5-taitettua sisältöä tablettija mobiilialustoille?*

Tutkimuskysymyksellä haluttiin toisaalta selvittää HTML5-taitetun sisällön asemaa eri sovellus- ja julkaisutyypeissä, mutta myös selvittää eri toteutusratkaisujen eroja muun muassa web-applikaation toteutuksessa.

- *Millä keinoilla voidaan parantaa HTML5-tekniikalla julkaistujen lehtien suorituskykyä?*

Pelkillä web-teknologioilla toteutetut ja selaimessa toimivat ratkaisut ovat perinteisesti olleet suorituskyvyllisesti altavastaajan asemassa natiiveihin sovelluksiin nähden. Tutkimuskysymyksellä haluttiin etsiä ratkaisuja suorituskyvyn parantamiseksi.

- *Mitkä ovat keskeisiä ongelmia ja rajoitteita HTML5-julkaisemisessa tabletti- ja mobiilialustoille? Millä keinoilla näitä ongelmia voidaan ratkaista?*

Viimeisen tutkimuskysymyksen kohdalla haluttiin paneutua julkaisemisen ongelmiin ja rajoitteisiin ja etsiä näille toimivia ratkaisuja. Lisäksi tavoitteena oli koota kaikkien edellä esitettyjen tutkimuskysymysten pohjalta suositeltujen käytäntöjen lista, jota voitaisiin hyödyntää HTML5-taitetun sisällön julkaisemisessa.

1.2 Aineisto ja tulokset

Keskeisenä aineistona diplomityössä käytettiin kahta demolehteä varten toteutettua sovellusta: natiivia Applen iOS-alustalle implementoitua wrapper-sovellusta, sekä diplomityön tuloksena syntyneen Stage Framework -sovelluskehyksen avulla toteutettua alustariippumatonta web-applikaatiota. Natiivi sovellus toteutettiin käyttämällä Baker eBook Framework -ohjelmistokehystä, joka tarjoaa Applen natiiveilta sovelluksilta vaatimalla Objective-C-kielellä ja Cocoa-ohjelmistoympäristöllä[1] kehitetyn sovelluspuitteen demolehden HTML5-sisällölle. Demolehden julkaisemiseksi saatiin siten kaksi vaihtoehtoa: alustariippuvainen natiivi sovellus, sekä alustariippumaton web-applikaatio.

Omana aineistonaan toimi myös itse demolehti, jonka sisältö koostui Aaltoyliopiston viestinnän julkaiseman Aalto University Magazinen järjestyksessään kolmannesta julkaistusta numerosta. Demolehti toteutettiin kokonaisuudessaan diplomityön aikana. Aalto University Magazinen eli AUM:n ja demolehden roolista kerrotaan tarkemmin jäljempänä, luvussa 1.3.

Diplomityön osana toteutettiin myös alusta- ja suorituskykytestaus, joka koostui keskeisiin teknologioihin liittyvien ominaisuuksien tuen kartoittamisesta ja teknologiakohtaisesta suorituskykytestauksesta. Tämä kvantitatiivinen osuus suoritettiin mobiilille digitaaliselle julkaisemiselle keskeisille Android, iOS ja Windows Phone -alustoille käyttäen erilaisia asiakaslaitteita. Alusta- ja suorituskykytestauksen tulokset palvelivat yhtenä keskeisenä aineistona tarjoten tietoa muun muassa alustojen asettamista rajoitteista ja käytettävissä olevien teknologioiden soveltuvuudesta. Testit suoritettiin myös yleisimmillä työpöytäselaimilla Apple OS X 10.8.2 ja Microsoft Windows 7 -käyttöjärjestelmillä.

Työn aineistona käytettiin myös muita tabletti- ja mobiilialustoille julkaistuja lehtiä, joista kartoitettiin muun muassa toteutustekniikoita ja käyttöliittymäratkaisuja. Erityisen kiinnostuneita diplomityössä oltiin HTML5-

[1] http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/CocoaFundamentals/

sisältöä käyttävistä lehdistä, joita diplomityön tekemisen alkaessa oli julkaistu niukasti. Toinen mielenkiintoinen segmentti koostui levikiltään isoista painetuista lehdistä - minkälaiseen ratkaisuun suuremmalla rahalla julkaistavat lehdet olivat päätyneet.

Diplomityön tuloksena syntyi toteutettujen sovelluksien, ohjelmistokehyksen, demolehden ja kvantitatiivisen osuuden lisäksi suositeltujen käytäntöjen lista, joka ohjeistaa HTML5-taitetun sisällön julkaisemiseen ja mahdollisen web-applikaation toteuttamiseen.

1.3 Taustaa

Diplomityön keskeisessä roolissa oli Aalto-yliopiston viestinnän yhteistyössä Alma 360:n[2] kanssa julkaisema sidosryhmälehti Aalto University Magazine, josta suunniteltiin ja toteutettiin demolehti tabletti- ja mobiilialustoille. Demolehden ulkoasun suunnitteli Juho Hiilivirta, joka on ollut toteuttamassa myös Aalto University Magazinen painettuja lehtiä. Ulkoasun suunnittelussa pyrittiin omalta osaltaan erottautumaan digitaalisesta näköislehdestä, joka on ollut luettavissa lehden verkkosivuilla[3]. Demolehti toteutettiin Juho Hiilivirran konseptikäsikirjan pohjalta yhteistyössä diplomityön ohjaajan, Mikko Kuhnan kanssa.

Demolehden sisältö koostui Aalto University Magazinen kolmannen numeron artikkeleista, joista suurin osa julkaistiin myös demolehdessä. Aalto University Magazinen painettu versio sisältää sekä suomen- että englanninkielisiä artikkeleita, joten myös demolehdestä haluttiin kaksikielinen versio. Aalto-yliopiston viestintä toimitti demolehteä varten kaksi kieliversiota suurimmasta osasta artikkeleita. Lisäksi lehteä varten toimitettiin audio- ja videosisältöä jakelukanavan tuomien mahdollisuuksien ansiosta, laajentaen painetun lehden sisältöä.

Aalto University Magazinen demolehti toteutettiin sisältöineen diplomityön ensimmäisessä vaiheessa, mutta lehteen tehtiin tarkennuksia työn edetessä Aalto-yliopiston viestinnän antaman palautteen perusteella. Iteroitu demolehti arvioitiin myöhemmässä vaiheessa käytettävyystutkimuksia suorittavan Adagen toimesta. Käytettävyysarvion pohjalta lehteen ei tehty enää suuria muutoksia, vaan ne jäävät aineistoksi Aalto-yliopiston viestinnälle, mikäli tabletti- ja mobiiliversiota päätetään alkaa julkaista painetun lehden ohella.

[2] http://alma360.fi/

[3] http://aalto.fi/fi/current/magazine/

Diplomityö on toteutettu osana Next Media -hanketta[4], josta myös kumpuaa tutkimuksellinen mielenkiinto alustariippumattomille julkaisuratkaisuille. Demolehden toteuttamisen yhteydessä on paneuduttu myös julkaisun automatisointiin liittyviin kysymyksiin, jotka ovat olleet keskiössä myös aikaisemmassa aiheeseen liittyvässä tutkimuksessa. Julkaisun automatisoinnista kerrotaan tarkemmin jäljempänä luvussa 5. Tärkeänä pohjana tämän diplomityön ohessa toteutetulle julkaisuautomaatiolle on kuitenkin toiminut Mikko Kuhnan suunnittelema algoritmi [1], joka etsii optimaalisimman rajauksen kuvalle halutulla kuvasuhteella käyttäen hyväksi kuvasta tunnistettavia tärkeitä ja mielenkiintoisia kohteita. Tärkeän informaation säilyttäminen kuvissa on avainasemassa suunniteltaessa HTML5-taitettujen lehtien julkaisemista eri kokoisille laitteille ja eri kuvasuhdetta oleville näytöille tai laiteorientaatioille.

[4] `http://www.nextmedia.fi/`

Luku 2

Digitaalinen julkaiseminen

Tässä luvussa käsitellään pääasiassa aikakaus- ja sanomalehtien julkaisemista tabletti- ja mobiilialustoilla, mutta sivutaan myös digitaalista julkaisemista yleisesti - jakelukanavien ja asiakaslaitteiden konvergenssin myötä raja tabletti- ja mobiilialustojen ja muiden alustojen välillä on muuttunut yhä häilyvämmäksi. HTML5 teknologiana edistää omalta osaltaan tätä konvergenssia, sillä sama sisältö voi mukautua helpommin ja luontevammin eri laitteille. Tämän diplomityön tekemisen aikana suurimmat yksittäiset tekijät tabletti- ja mobiilialustasegmentin erottautumiselle olivat vahvat mobiilikäyttöjärjestelmät ja niiden kauppapaikat, joissa natiivit sovellukset pitävät edelleen pintansa.

2.1 Ympäristö

Tässä alaluvussa käydään lävitse digitaaliseen julkaisemiseen vaikuttavia trendejä, itse alustat ja digitaaliset kauppapaikat esitellään tarkemmin luvussa 2.3. Vuoden 2012 aikana Android-pohjaiset laitteet hallitsivat myytyjen laitteiden markkinaosuutta. Seuraavassa taulukossa 2.1 on esitetty tutkimus- ja konsultointiyritys Gartnerin julkaisema tilasto vuosien 2011 ja 2012 kahden viimeisen vuosineljänneksen markkinaosuuksista alustakohtaisesti [2, 3].

Taulukko 2.1: Maailmanlaajuinen mobiililaitteiden myynti vuosien 2012[5] ja 2011[6] kahdella viimeisellä neljänneksellä.

Alustat	**Q4 2012**	**Q3 2012**	**Q4 2011**	**Q3 2011**
Android	69,7%	72,4%	51,3%	52,5%
iOS	20,9%	13,9%	23,6%	15,0%
Research In Motion	3,5%	5,3%	8,8%	11,0%
Microsoft	3,0%	2,4%	1,8%	1,5%
Bada	1,3%	3,0%	2,1%	2,2%
Symbian	1,2%	2,6%	11,6%	16,9%
Muut	0,3%	0,4%	0,8%	0,9%
Yhteensä	100,0%	100,0%	100,0%	100,0%

Kuten myös Gartner kolmannen vuosineljänneksen raportissaan huomauttaa, on taulukossa esitetyissä markkinaosuuksissa otettava huomioon vuoden 2012 viimeisellä vuosineljänneksellä tapahtunut iPhone 5 -puhelimen lanseeraus, joka omalta osaltaan on vaikuttanut iOS-laitteiden myynnin hiljentymiseen kolmannella vuosineljänneksellä [2]. Windows 8 ja Windows Phone 8 -käyttöjärjestelmien julkaiseminen vuoden 2012 viimeisellä neljänneksellä vaikuttivat Microsoftin markkinaosuuteen myönteisesti nostaen Microsoftin Badan ja Symbianin ohitse neljänneksi myydyimmäksi. Useiden tabletti- ja mobiililaitteiden julkaiseminen uusilla käyttöjärjestelmillä vuoden 2012 ja 2013 aikana on Microsoftin ensimmäisiä todellisia yrityksiä päästä mukaan tabletti- ja mobiilimarkkinoille. Huolimatta Android-laitteiden vahvasta myynnistä useat eri alustojen markkinaosuutta verkossa tarkkailevat palvelut raportoivat Applen iOS-laitteiden hallitsevan verkon käyttöä mitattaessa. Usein referoidun Net Applications -palvelun mukaan joulukuussa 2012 iOS-laitteet hallitsivat verkkokäyttöä 60,1%:n osuudella verrattuna Android-laitteiden 24,6%:n osuuteen tabletti- ja mobiililaitteiden segmentissä[7].

HTML5-julkaisemisen kannalta mielenkiintoista on Android ja iOS -alustojen tukeutuminen WebKit-selainmoottoriin käyttöjärjestelmien oletusselaimissa. Tämä yhtenäistää ja helpottaa web-pohjaista kehitystä näille alustoille. Selaimia ja niiden renderöintimoottoreiden roolia käsitellään tarkemmin luvussa 2.4.3. WebKitin osuus tabletti- ja mobiilimarkkinoilla on kuitenkin millä tahansa mittarilla mitattuna huomattava. Diplomityön kirjoittamisen

[5] http://gartner.com/newsroom/id/2335616

[6] http://gartner.com/newsroom/id/2237315

[7] http://netmarketshare.com/operating-system-market-share.aspx?qprid=9&qpcustomb=1

aikana Applen iOS-alustalle ei ole saatavilla muita selainmoottoreita. Kaikki alustalle julkaistut selaimet käyttävät käyttöjärjestelmän mukana toimitettua selainmoottoria [4]. Androidille on julkaistu versiot kaikista yleisimmistä selaimista omilla selainmoottoreillaan, jotka käyttäjä voi asentaa Google Play -palvelusta tai Amazon Appstoresta Kindle-laitteille.

Mobiileilla alustoilla uusien teknologioiden adoptointiin eivät vaikuta ainoastaan laitteiden ja alustojen kehittäjien aktiivisuus, vaan myös potentiaali virallisten ohjelmistopäivitysten nopeaan lanseeraamiseen, mihin omalta osaltaan vaikuttaa käyttäjien aktiivisuus ja päivityksen helppous. Android-käyttöjärjestelmän aktiivisten versioiden määrä markkinoilla on suurempi muun muassa laitekannan suorituskykyerojen vuoksi - vanhemmat laitteet eivät tue enää uudempia käyttöjärjestelmäversioita. Toisaalta iOS-laitteilla päivitysten adoptointinopeus on markkinoiden vilkkainta. Taulukoissa 2.2 ja 2.3 on esitetty eri ohjelmistoversioiden osuudet sekä Android että iOS -alustoilla joulukuussa 2012.

Taulukko 2.2: Google Play -sovelluskaupassa vierailleiden laitteiden ohjelmistoversiot 14 päivän tarkkailujaksolla päättyen 3. joulukuuta 2012.[8]

Versio	Koodinimi	Rajapintaversio	Markkinaosuus
1.5	Cupcake	3	0,1%
1.6	Donut	4	0,3%
2.1	Eclair	7	2,7%
2.2	Froyo	8	10,3%
2.3 - 2.3.2	Gingerbread	9	0,2%
2.3.3 - 2.3.7	Gingerbread	10	50,6%
3.1	Honeycomb	12	0,4%
3.2	Honeycomb	13	1,2%
4.0.3 - 4.0.4	Ice Cream Sandwitch	15	27,5%
4.1	Jelly Bean	16	5,9%
4.2	Jelly Bean	17	0,8%

Apple ei julkaise iOS-alustasta virallisia versioiden markkinaosuuksia, mutta erilaisia sovelluskehittäjien ylläpitämiä tilastoja on saatavilla. Yksi usein viitattu lähde on Audiobooks-sovelluksen laitekanta. Sovellus ei tue ohjelmistoversioita, jotka ovat vanhempia kuin 4.3.0, mutta niiden osuus oli tuen pu-

[8] `http://developer.android.com/about/dashboards/`

dottamishetkellä 4%:n luokkaa[9]. Seuraavan taulukon tiedot on kerätty 28. joulukuuta 2012.

Taulukko 2.3: David Smithin Audiobooks-sovelluksen laitekannan iOS-ohjelmistoversiot 28. joulukuuta 2012.[9]

Versio	Osuus laitteista	Julkaisupäivämäärä
6.0.x	79,2%	19. syyskuuta 2012
5.1.x	15,4%	7. maaliskuuta 2012
5.0.x	2,0%	12. lokakuuta 2011
4.3.x	2,6%	9. maaliskuuta 2011
4.2.x	0,6%	22. marraskuuta 2010
4.1.x	0,2%	8. syyskuuta 2010

Mikäli tarkastellaan digitaalista julkaisemista yleisesti, on otettava huomioon myös työpöytäjärjestelmät, jotka edustavat edelleen merkittävää osaa digitaalisen sisällön kuluttamiseen soveltuvista alustoista. Riippuen lähteestä työpöytäselainten markkinaosuudet vaihtelevat erityisesti Microsoftin Internet Explorer -selainten osalta. Taulukossa 2.4 on esitetty verkon standardeja julkaisevan ja ylläpitävän World Wide Web Consortiumin, W3C:n[10] W3Schools-sivuston tilasto marraskuulta 2012 ja StatCounterin tiedot joulukuulta 2012.

Taulukko 2.4: Selainten markkinaosuudet W3Schools-sivustolla marraskuussa 2012[11] ja StatCounterilla joulukuussa 2012[12].

Selain	W3Schools	StatCounter
Chrome	46,3%	36,42%
Firefox	31,2%	21,89%
Internet Explorer	15,1%	30,78%
Safari	4,4%	7,92%
Opera	2,0%	1,26%
Muut	1,0%	1,73%

[9] http://david-smith.org/iosversionstats/

[10] http://w3.org/

[11] http://w3schools.com/browsers/browsers_stats.asp

[12] http://gs.statcounter.com/

2.2 Ansaintamallit

Käytännöt rahavirtojen saamiseksi digitaalisten julkaisujen avulla vaihtelevat suuresti. Kuitenkin perinteinen ansaintamalli on keskittynyt mainostilan myymiselle samalla tavalla kuin painetuissa lehdissäkin, nyt vain digitaalisena ja mainonnan tehokkuuteen perustuvalla hinnoittelulla: mainostajat maksavat esimerkiksi mainosten näyttökertojen määrästä tai kohdepalveluun mainosten kautta päätyvistä käyttäjistä [5]. Digitaalisia mainoksia voidaan kohdentaa käyttäjän sijainnin mukaan siinä missä painetun lehdenkin, mutta entistä tarkemmin. Mainoksia voidaan kohdentaa myös muiden demografisten tekijöiden mukaan, jos palvelu tällaisia kerää, esimerkiksi rekisteröityneiltä käyttäjiltä - kuten lehden tilaajilta. Gartnerin ennusteen mukaan mobiilimainonnan vuosittainen liikevaihto kehittyisi 24,5 miljardiin dollariin vuoteen 2016 mennessä vuoden 2012 alle kymmenestä miljardista dollarista [6].

Tabletti- ja mobiilialustoilla mukaan ovat tulleet entistä vahvemmin keskitetyt mainosalustat, kuten Apple iAd[13] ja Google AdSense[14]. Näillä alustoilla alustan ylläpitäjä hallinnoi näytettäviä mainoksia ja tarjoaa mainostuloja mainosten julkaisijoille, sekä vastaavasti myy mainostilaa mainostajille [7]. Mainosten julkaisijalla on kuitenkin hyvin vähän mahdollisuuksia vaikuttaa mainosten esitystapaan tai sisältöön, mikä tekee niistä arvaamattomia digitaalisille julkaisuille. Dynaamisesti generoitu mainossisältö aiheuttaa myös toisenlaisen ongelman lehden lukijalle: mikäli mainos vaihtuu sivulta toiselle selattaessa, ei käyttäjä välttämättä löydä enää aikaisemmin näkemäänsä mainosta. Mainonnasta katoaa myös painetulle julkaisulle tyypillinen pysyvyys - kukaan ei voi ikinä palata katsomaan, mitä jonkin lehden jossakin numerossa mainostettiin kymmenen vuotta ajassa taaksepäin.

Mainokset ovat kuitenkin vain yksi ansaintamalli digitaalisessa julkaisemisessa. Alustakohtaiset natiivit sovellukset ja niiden kauppapaikat tarjoavat mahdollisuuden lehtien julkaisemiseen yhdessä sovelluksessa tai numerokohtaisesti erikseen useammassa sovelluksessa, mikäli kyseessä on harvemmin ilmestyvä julkaisu. Julkaisija voi kerätä tuloja myymällä itse sovellusta tai tarjoamalla sovelluksen ilmaiseksi ja tukeutumalla sovelluksen sisäisiin ansaintamalleihin [8].

Mainonnan lisäksi suosittuja ovat sovelluksen sisäiset ostokset, joissa käyttäjä ostaa sisältöä sovelluksen sisällä [8]. Oleellista ei kuitenkaan ole missä itse ostotapahtuma tapahtuu, sillä monet lehdet tarjoavat painetun lehden tilaa-

[13] `http://advertising.apple.com/`

[14] `http://google.com/adsense/`

jilleen digitaalisen lehden ilmaiseksi. Digitaalisen lehden lukija saa käyttöönsä tunnistautumalla sovelluksen sisällä käyttäjätunnuksensa tai asiakasnumeron ja salasanan yhdistelmällä. Jatkuvan tai määräaikaisen tilauksen mahdollistaminen sovelluksen sisällä helpottaa kuitenkin käyttäjän ostopäätöksen toteuttamista. Käyttäjille voidaan tarjota myös mahdollisuutta ostaa julkaisuja numerokohtaisesti, jolloin tuloja saadaan kerättyä todennäköisemmin myös satunnaisilta lukijoilta.

Diplomityössä kehitetyssä ohjelmistokehyksessä ei oteta kantaa käytettyyn ansaintamalliin. Ohjelmistokehyksen web-applikaatio sisältää lehtihyllyn, jonka yhteyteen voidaan jatkokehityksellä yhdistää käyttäjien autentikointi ja kauppapaikka. Lehtien julkaisija voi myös valita mainosten tai täysin ilmaisen jakelun väliltä.

2.3 Alustat ja digitaaliset kauppapaikat

Alustat voidaan perinteisessä mielessä jakaa mobiili- ja työpöytäkäyttöjärjestelmiin. Mobiilikäyttöjärjestelmällä viitataan tässä yhteydessä ajalle tyypillisesti ARM-suoritinarkkitehtuuria tukevaan käyttöjärjestelmään erotuksena työpöytäkäyttöjärjestelmien x86-arkkitehtuurista. Eri arkkitehtuureille suunniteltuja ohjelmistoja ja sovelluksia ei voida ilman ylimääräistä työtä suorittaa toisen arkkitehtuurin laitteistolla. Tulevaisuus on kuitenkin avoin hybridikäyttöjärjestelmille, jotka pystyvät suorittamaan molempien arkkitehtuurien koodia, esimerkiksi reaaliaikaisella binäärikääntäjällä. Tähän perustui muun muassa Rosetta[15], jolla Apple käänsi PowerPC:lle suunniteltujen ohjelmien koodia reaaliaikaisesti Intelin x86-arkkitehtuurille OS X -käyttöjärjestelmällä, Applen siirryttyä PowerPC-suorittimista Intelin suorittimiin.

2.3.1 Android

Laitemäärältään suurin mobiilikäyttöjärjestelmä on Googlen kehittämä Android, jota Google lisensoi useille eri laitevalmistajille, kuten Samsungille ja HTC:lle. Laitevalmistajat voivat tehdä muutoksia käyttöjärjestelmään - esimerkiksi HTC käytää omaa HTC Sense -käyttöliittymää. Amazon puolestaan tarjoaa Android-käyttöjärjestelmällä varustettuja Kindle-tabletteja, joiden käyttöliittymää on niin ikään mukautettu. Lisäksi Kindle-laitteissa oletuksena tulevat muun muassa kirjojen ja lehtien kauppapaikka Kindle for Android, sekä Amazon Appstore, vaihtoehtoinen sovelluskauppa Google Play

[15] `http://apple.com/asia/rosetta/`

-kaupalle, joka on Androidin oletuskauppapaikka muiden valmistajien laitteilla. Kindle for Android -sovellus on ladattavissa myös muille Android-laitteille[16]. Suurin Android-laitevalmistaja on Samsung, jonka lippulaivoina toimivat Galaxy S -sarjan puhelimet kilpailevat myyntiluvuissa Applen iPhone -puhelimia vastaan [9].

Yhtenä Android-mobiilikäyttöjärjestelmän ongelmana voidaan sovelluskehityksen ja digitaalisen julkaisemisen kannalta pitää myös aikaisemmin esitellystä taulukosta 2.2 nähtävää pirstaloitunutta laitekantaa, joka kattaa suuren määrän eri ohjelmistoversioita ja suorituskyvyltään vaihtelevia laitteita. Esimerkiksi kiinalainen Huawei on yksinomaan keskittynyt edullisempien tuotesegmenttien Android-laitteisiin.

Sovelluskehittäjät voivat julkaista natiiveja sovelluksiaan esimerkiksi Google Play -kauppapaikassa tai tarjota selaimella käytettäviä web-applikaatioita. Android-laitteilla käyttäjä ei pysty lisäämään web-applikaatiota muiden sovellusikonien joukkoon kotinäytölle tai sovellusvalikkoon.

2.3.2 iOS

iOS on Applen kehittämä ja yksinoikeudellinen mobiilikäyttöjärjestelmä, jota käytetään iPhone, iPad, iPod Touch ja Apple TV -laitteissa. Alustan kauppapaikkana toimii Applen ylläpitämä App Store, joka on tunnettu tarkoista sovelluksia koskevista vaatimuksista. App Storeen kytkeytyy Lehtikioski (engl. Newsstand), joka on erillinen digitaalisten lehtien tilaamiseen ja hallintaan tarkoitettu sovellus. Käyttäjät voivat selailla lehtitarjontaa App Storen Lehtikioski-kategoriassa ja asentaa haluamansa lehdet Lehtikioskiin nopeasti saataville. Käytännössä lehdet ovat iOS-sovelluksia, joiden toteutustapa on vapaa sovelluksilta vaadittujen vaatimusten rajoissa. Sovelluksen toiminta Newsstand-sovelluksena määritellään Newsstand Kit ja Store Kit -ohjelmistokehysten avulla, jotka ovat osa iOS-kehitystyökaluja[17].

Kirjoille ja muille julkaisuille Apple tarjoaa iBooks-sovelluksen, jolle kehitetään omalla, OS X:llä toimivalla iBooks Author -sovelluksella[18]. iBooks Authorilla tehdyt kirjat pääsevät ilmaiseen jakeluun iBookstoreen, mutta ainoastaan ISBN-koodin saaneita kirjoja voidaan myydä maksua vastaan. Myöskään painetun kirjan ISBN-koodia ei voida käyttää samansisältöiselle digitaaliselle kirjalle[19].

[16] http://amazon.com/gp/feature.html?docId=165849822

[17] http://developer.apple.com/devcenter/ios/newsstand/

[18] http://apple.com/ibooks-author/

[19] http://apple.com/itunes/content-providers/book-faq.html

2.3.3 Windows Phone ja Windows RT

Windows Phone on Microsoftin kehittämä mobiilikäyttöjärjestelmä, jota Microsoft lisensoi laitevalmistajille, kuten HTC:lle ja Nokialle. Windows Phone eroaa merkittävästi Microsoftin Windows-työpöytäkäyttöjärjestelmästä, josta julkaistiin 25. lokakuuta 2012 Windows 8 ja sen ARM-arkkitehtuurin suorittimilla toimiva versio Windows RT[20]. Microsoft julkaisi samassa yhteydessä Surface-tuotesarjan, joka koostuu Microsoftin omista tablettilaitteista: Surface RT:stä ja Surface Pro:sta. Nimensä mukaisesti tableteissa on käyttöjärjestelminä Windows RT ja Windows 8[21]. Windows RT -laitteet voivat ajaa yksinomaan Windows Storesta ostettuja sovelluksia, kun taas Windows 8 Pro -laitteet tukevat kaikkia saatavilla olevia Windows-sovelluksia[22]. Windows Phone -mobiilikäyttöjärjestelmän kauppapaikkana toimii Windows Phone Store.

2.3.4 Muut mobiilikäyttöjärjestelmät

Muiden mobiilikäyttöjärjestelmien osuus markkinoista on vaihdellut markkina-alueesta riippuen merkittävästikin, esimerkiksi kanadalainen RIM eli Research In Motion[23] on pitänyt kiinni markkinaosuudestaan paremmin Pohjois-Amerikassa, vaikka sekin on joutunut taipumaan iOS:n ja Androidin kasvattaessa osuuksiaan. RIM on viime vuosina tarjonnut kahta erillistä mobiilikäyttöjärjestelmää eri laitetyypeille: BlackBerry OS -käyttöjärjestelmää yhtiön BlackBerry-puhelimille ja BlackBerry Tablet OS -käyttöjärjestelmää PlayBook-tableteille. RIMin tuleva BlackBerry 10 -mobiilikäyttöjärjestelmän on tarkoitus yhdistää nämä erilliset käyttöjärjestelmät.

Nokian läsnäolo mobiilimarkkinoilla pohjautuu yhtiön strategian mukaan yksinomaan Microsoftilta lisensoitaville käyttöjärjestelmille, kuten Windows Phone 7.5:lle ja 8:lle. Yhtiön vanha käyttöjärjestelmä Symbian OS[24] on erillisinä S40 ja S60 -versioina edelleen mukana yhtiön tuoteportfoliossa, mutta väistyvänä käyttöjärjestelmänä. Symbianin markkinaosuus onkin nopeasti pienentynyt maailmanlaajuisesti hallitsevasta mobiilikäyttöjärjestelmästä marginaaliselle tasolle.

Nokia oli aikaisemmin mukana kehittämässä MeeGo-mobiilikäyttöjärjestel-

[20] http://microsoft.com/en-us/news/Press/2012/Oct12/10-25Windows8GAPR.aspx
[21] http://microsoft.com/Surface/en-US/
[22] http://microsoft.com/Surface/en-US/surface-with-windows-rt/help-me-choose/
[23] http://rim.com/
[24] http://developer.nokia.com/Community/Wiki/Symbian_OS

mää[25], joka perustui Androidin tapaan Linux-käyttöjärjestelmään yhdistäen Intelin Mobli[26] ja Nokian Maemo[27] -käyttöjärjestelmät. Nokian vetäydyttyä aktiivisen kehittäjän roolista MeeGo on siirtynyt muun muassa Intelin, Linux Foundationin ja Samsungin aktiivisempaan kehitykseen uutena Tizen-käyttöjärjestelmänä[28]. Maailman suurin matkapuhelinvalmistaja Samsung on keskittynyt pääasiassa Android-käyttöjärjestelmään mobiililaitteissaan, mutta valmistaa siis myös muilla alustoilla varustettuja laitteita. Yksi näistä on Samsungin oma mobiilikäyttöjärjestelmä bada[29], joka muistuttaa ulkoasultaan Samsungin käyttämän Androidin käyttöliittymää. Huolimatta pienestä markkinaosuudesta badalla varustettuja laitteita on julkaistu runsaasti.

Muita tuloillaan olevia mobiilikäyttöjärjestelmiä ovat suomalaisen Jollan kehittämä Android-yhteensopiva Sailfish OS[30], Mozillan Firefox OS[31], sekä Canonical-yrityksen kehittämät Ubuntu Phone[32] ja Ubuntu for ARM[33].

2.3.5 Työpöytäkäyttöjärjestelmät

Työpöytäkäyttöjärjestelmistä markkinosuudeltaan suurin on Microsoftin kehittämä Windows, josta on markkinoilla suurilla markkinaosuuksilla useita eri versioita. Näistä vanhin merkittävällä osuudella on XP, joka julkaistiin vuonna 2001. Microsoftin uusimmat Windows-versiot, 8 ja RT tukevat Windows Store -kauppapaikkaa, josta käyttäjät voivat ladata alustan tukemia sovelluksia: RT-käyttöjärjestelmällä ainoastaan ARM-arkkitehtuurille kehitettyjä sovelluksia.

Apple tarjoaa omilla Mac-laitteillaan toimivaa OS X -käyttöjärjestelmää, jonka markkinaosuutta iOS:n suosio on omalta osaltaan kasvattanut viime vuosien aikana. OS X -käyttöjärjestelmä tukee omaa Mac App Store -kauppapaikkaa, josta käyttäjät voivat ostaa OS X:lle kehitettyjä sovelluksia. Mac App Store on hyvin samankaltainen iOS:n App Storen kanssa. Apple kerää Mac App Storessa myytyjen sovellusten tuloista iOS:n tapaan kiinteän prosenttiosuuden itselleen.

Kolmas ja viimeinen digitaalisen julkaisun kannalta merkittävä työpöytäkäyt-

[25] http://meego.com/
[26] http://mobli.com/
[27] http://maemo.org/
[28] http://tizen.org/
[29] http://bada.com/
[30] http://sailfishos.org/
[31] http://mozilla.org/firefoxos/
[32] http://ubuntu.com/devices/phone/
[33] http://ubuntu.com/download/arm/

töjärjestelmä on Linux, jonka jakeluversioista yleisin tavallisten käyttäjien käytössä on Canonicalin kehittämä Ubuntu[34]. Ubuntu sisältää Ubuntu Software Center -sovelluksen, josta käyttäjät voivat etsiä ja ladata käyttöjärjestelmän tukemia sovelluksia. Ubuntu Software Center sisältää Mac App Storen ja Windows Storen tapaan myös maksullisia sovelluksia.

2.4 Toteutusvaihtoehdot ja -ympäristöt

Niin sovelluskehityksessä kuin digitaalisten lehtien julkaisemisessa tablettija mobiilialustioille on olemassa erilaisia toteutustapoja ja -tekniikoita. Yksi keskeisimmistä valinnoista on sovellus- ja julkaisutyypin valinta. Riippuen valitusta toteutustekniikasta digitaalisen julkaisun sovelluksen kehittäjät joutuvat tekemisiin joko alustakohtaisten kehitysympäristöjen ja -tekniikoiden tai selainteknologioiden kanssa. Tässä alaluvussa käydään lävitse eri sovellus- ja julkaisutyypit, alustakohtaiset kehitysympäristöt, sekä selain- ja renderöintimoottorit.

2.4.1 Sovellus- ja julkaisutyypit

Sovellustyypit jaetaan tyypillisesti toteutustavan mukaan kolmeen kategoriaan: natiiviin, hybridi-sovellukseen ja web-applikaatioon. Digitaalisen julkaisemisen kannalta jako on kuitenkin pelkistetty, joten tässä diplomityössä näiden sovellustyyppien lisäksi on esitelty erikseen wrapper-sovellus ja muut julkaisutyypit, johon lukeutuvat muun muassa alustakohtaiset erikoisjulkaisut, verkkojulkaisut ja näköislehdet.

2.4.1.1 Natiivit sovellukset

Natiiveilla sovelluksilla tarkoitetaan erityisesti mobiilikäyttöjärjestelmien kohdalla alustakohtaisia sovelluksia, jotka käyttävät alustan tarjoamia ohjelmointirajapintoja, kuten käyttöliittymäkirjastoja. Natiivien sovelluksien etuna on lähtökohtaisesti parempi suorituskyky, sillä ne voivat hyödyntää laitteistojen tarjoamia resursseja, kuten rautapohjaista grafiikkakiihdytystä täysipainoisesti. Käyttääkseen natiivia sovellusta käyttäjän on ensin asennettava se tyypillisesti mobiilikäyttöjärjestelmän omasta keskitetystä kauppapaikasta. Natiiveilla sovelluksilla on myös mahdollisuus tallentaa tietoja

[34] http://ubuntu.com/

laitteen muistiin niille varattuun tallennustilaan. Pääsy laitteen tiedostojärjestelmään on usein rajoitettu (engl. sandboxed). Myös koko sovelluksen toiminta voidaan erottaa muusta käyttöjärjestelmässä esimerkiksi virtualisoimalla, kuten Androidissa[35]. Applen App Store oli ensimmäinen alustakohtainen kauppapaikka, joka onnistui haastamaan perinteisen operaattorivetoisen mobiilisisällön ja -palveluiden tarjoamisen [10].

2.4.1.2 Hybridisovellukset

Hybridisovellukset ovat natiiveja sovelluksia, jotka tukeutuvat sisällössään HTML:ään. Hybridisovelluksille tyypillistä on natiivien käyttöliittymäelementtien käyttö, jotta sovellus vastaisi käyttötuntumaltaan täysin natiivia sovellusta. Hybridisovellukset voivat joko käyttää paikallisia web-resursseja HTML-sisällössään tai ladata tietoja palvelimelta. Tyypillisiä palvelimelta ladattavaan tietoon tukeutuvia hybridisovelluksia ovat erilaiset sosiaalisen median sovellukset, kuten Foursquare[36]. Hybridisovelluksien tekemiseen on myös olemassa täysin räätälöityjä ohjelmistokehyksiä, kuten luvussa 3.4.8 mainittu PhoneGap, jolla natiivien käyttöliittymäelementtien käyttäminen muuten HTML-muotoisessa sisällössä onnistuu ohjelmistokehyksen avulla [11].

2.4.1.3 Wrapper-sovellukset

Wrapper-sovellukset on eroteltu tässä diplomityössä omaksi julkaisutyypikseen erotuksena hybridisovelluksista, sillä ne eivät tarvitse natiiveja käyttöliittymäkirjastoja, eivätkä yritä tekeytyä ulkoasultaan natiiviksi sovellukseksi. Esimerkkinä wrapper-sovelluksesta toimii diplomityössä toteutettu iOS-sovellus, joka käyttää luvussa 3.4.2 esiteltyä Baker eBook Framework -ohjelmistokehystä. Sovelluskehys on toteutettu käyttäen Objective-C -kieltä ja iOS SDK:n rajapintoja. Kuitenkin sisältö toteutetaan yksinomaan HTML5:n avulla navigaatiota myöten. Bakerin tapauksessa kuitenkin esimerkiksi sisältöjen lataamiseen ja sivujen pyyhkäisyeleen toteuttamiseen on käytetty iOS:n natiiveja kirjastoja.

2.4.1.4 Web-applikaatiot

Web-applikaatiot nähdään tyypillisesti vastakohtana natiiveille sovelluksille, vaikka ne pyrkivät viime kädessä pääsemään mahdollisimman lähelle natiivin

[35] `http://developer.android.com/training/articles/security-tips.html`
[36] `http://foursquare.com/`

sovelluksen toimintaa niin käyttöliittymän toiminnassa kuin suorituskyvyssä. Applen iOS-alustalla on erityisesti panostettu web-applikaatioiden integroimiseen natiivien sovellusten joukkoon tarjoamalla esimerkiksi sovelluksen lisäämisen koti-valikkoon ja toteuttamalla rajapinnat sovelluksen aloituskuvien ja ikonien määrittämiseksi web-applikaatiolle. Muilla alustoilla web-applikaatiot toimivat pääasiallisesti selaimessa. HTML5 on tuonut mukanaan ominaisuuksia, jotka ovat parantaneet erityisesti web-applikaatioiden asemaa. Näihin ominaisuuksiin kuuluvat muun muassa HTML5-sovellusvälimuisti ja HTML5-tallennustila. Myös HTML5-piirtoalue ja WebGL-rajapinta reaaliaikaiselle 3D-grafiikalle voidaan nähdä web-applikaatioiden mahdollisuutena. Spesifikaatioissa määritellään myös vielä yleistymättömiä rajapintoja selaimille, jotka mahdollistavat web-applikaatioille entistä paremman pääsyn laitteen toiminnallisuuksiin, kuten kameroihin ja tiedostojärjestelmään. HTML5 on esitelty tarkemmin luvussa 3.1.1.

2.4.1.5 Muut julkaisutyypit

Muita mahdollisia julkaisutyyppejä ovat esimerkiksi alustakohtaiset erityisasemassa olevat julkaisut, kuten Applen iBooks Authorilla tehdyt julkaisut, joita voi hankkia ja lukea iOS-alustan iBooks-sovelluksen avulla. iBooks Author tarjoaa valmiita julkaisupohjia, joissa voidaan käyttää muun muassa kolmiulotteisia kuvia, videota ja animaatioita. iBooks Authorilla tehdyt julkaisut perustuvat osittain EPUB-formaattiin[37].

Yksi mahdollinen julkaisutyyppi on verkkojulkaisu, joka perustuu verkkosivustoon ja mahdollisesti sen laitekohtaiseen optimointiin esimerkiksi responsiivisella suunnittelulla tai tukemalla pikselitiheydeltään tarkempia näyttöjä tarjoamalla niille tarkempia kuvia.

Yksi edelleen paljon käytetty julkaisutyyppi on näköislehti, joka on helposti tuotettavissa, mutta ei tarjoa lisäsisältöä tai erityistä tukea jollekin laitetyypille. Käytännöt näköislehden jakelutavasta vaihtelevat suuresti natiiveista sovelluksista aina PDF-tiedostoformaattiin[38].

2.4.2 Alustojen kehitysympäristöt

Android tarjoaa kehitystyökaluina oman SDK:n, joka on paketoitu yhteen ADT:n (Android Developer Tools) kanssa. Paketissa tulee mukana Eclipse-

[37] http://support.apple.com/kb/HT4059

[38] http://adobe.com/products/acrobat/adobepdf.html

editori, jossa on oma plugin ADT:lle. Android-kehitystä varten on saatavilla BlueStacks, jolla Android-sovelluksia voidaan ajaa suoraan OS X tai Windows -käyttöjärjestelmillä[39]. Sovellus toimii huomattavasti kevyemmin kuin esimerkiksi Android SDK:n mukana tuleva emulaattori, jolla voi luoda virtuaalikoneita eri laiteprofiileilla. Emulaattori tulkkaa ARM-koodia lennosta x86-arkkitehtuurille, mikä osaltaan hidastaa sen toimintaa.

iOS:n kehitysympäristönä toimii Apple OS X:llä toimiva Xcode, jota käytetään yhdessä iOS SDK:n kanssa. Käytettävä ohjelmointikieli on Objective-C, joka on Applen oma C-kieltä laajentava kieli. Xcode ja iOS SDK ovat saatavilla ainoastaan OS X:lle, joten kehittäjät tarvitsevat Mac-tietokoneet kehitystä varten. Xcoden mukana tulee iOS Simulator -sovellus, joka ajaa kehitetyn sovelluskoodin natiivisti x86-arkkitehtuurilla. Simulaattori on todella nopea, mutta voi joissain tapauksissa käyttäytyä eri tavoin oikeassa laitteessa suoritettavaan sovellukseen verrattuna. Tämä voi johtua muun muassa eroista muistin ja käytettävän levytilan määrässä sekä sovelluksen käytettävissä olevasta suoritintehosta.

Windows RT käyttää yhteistä kehityspakettia Windows 8:n kanssa: Windows Software Development Kit (SDK) for Windows 8. Windows Phone -käyttöjärjestelmälle on oma Windows Phone SDK, joka eroaa Windowsin pöytä- ja tablettikäyttöjärjestelmien SDK:sta.

2.4.3 Selain- ja renderöintimoottorit

Kuten web-kehityksessä yleensä, on myös HTML5-julkaisemisessa otettava huomioon selainten ja selainmoottoreiden väliset eroavaisuudet. Vaikka WebKit on ylivoimaisesti käytetyin selain tabletti- ja mobiilialustoilla, ovat Microsoftin Trident-selaimet kiinteä osa Windows ja Windows Phone -alustoja. Myös esimerkiksi Gecko ja Presto ovat saatavilla preinteisten x86-järjestelmien lisäksi muun muassa Androidille. Yleisimmät selaimet ja niiden käyttämät selainmoottorit on esitetty seuraavassa taulukossa 2.5.

[39] `http://bluestacks.com/`

Taulukko 2.5: Yleisimpien selaimien kehittäjät, selainmoottorit ja niiden ensisijaiset JavaScript-moottorit.

Selain	Kehittäjä	Selainmoottori	JavaScript-moottori
Android Browser	Google	WebKit	V8
Chrome	Google	WebKit	V8
Chromium	Yhteisö	WebKit	V8
Dolphin (HD)	MoboTap	WebKit	V8
Firefox	Mozilla, Mozilla Foundation	Gecko	JägerMonkey
Internet Explorer	Microsoft	Trident	Chakra
Konqueror	Yhteisö, KDE	KHTML / WebKit	KJS
Maxthon	Maxthon International	WebKit / Trident	V8
Opera (Mobile)	Opera Software	Presto	Carakan
Safari	Apple	WebKit	Nitro

iOS:llä selaimet käyttävät tietoturvasyistä samaa WebKit-selainmoottoria, jota Safari käyttää. Myös Microsoft on rajoittanut Windows RT -mobiilikäyttöjärjestelmässä kolmansien osapuolten sovellusten pääsyä perinteiseen Win32-rajapintaan samasta syystä, mikä käytännössä tekee mahdottomaksi kilpailun alustan oletusselaimen, Internet Explorerin kanssa[40]. Internet Explorerin mobiiliversio on kymmenennen version myötä ottanut ensimmäisiä kunnon askeliaan ottaakseen muut selaimet kiinni moderneissa selainteknologioissa. Internet Explorerin mobiiliversion ilmoitetaan käyttävän samaa selainmoottoria työpöytäkäytössä olevan IE 10:n kanssa[41].

[40] http://extremetech.com/computing/129137-microsoft-bans-third-party-browsers-from-windows-on-arm/

[41] http://blogs.windows.com/windows_phone/b/windowsphone/archive/2012/06/20/announcing-windows-phone-8.aspx

Luku 3

Tekniikat ja teknologiat

Tässä luvussa käsitellään diplomityössä käytettyjä tekniikoita ja teknologioita spesifikaatioista suunnitteluparadigmojen ja suorituskykyohjeistojen kautta aina toteutetuissa sovelluksissa käytettyihin ohjelmointikehyksiin ja -kirjastoihin.

3.1 Spesifikaatiot

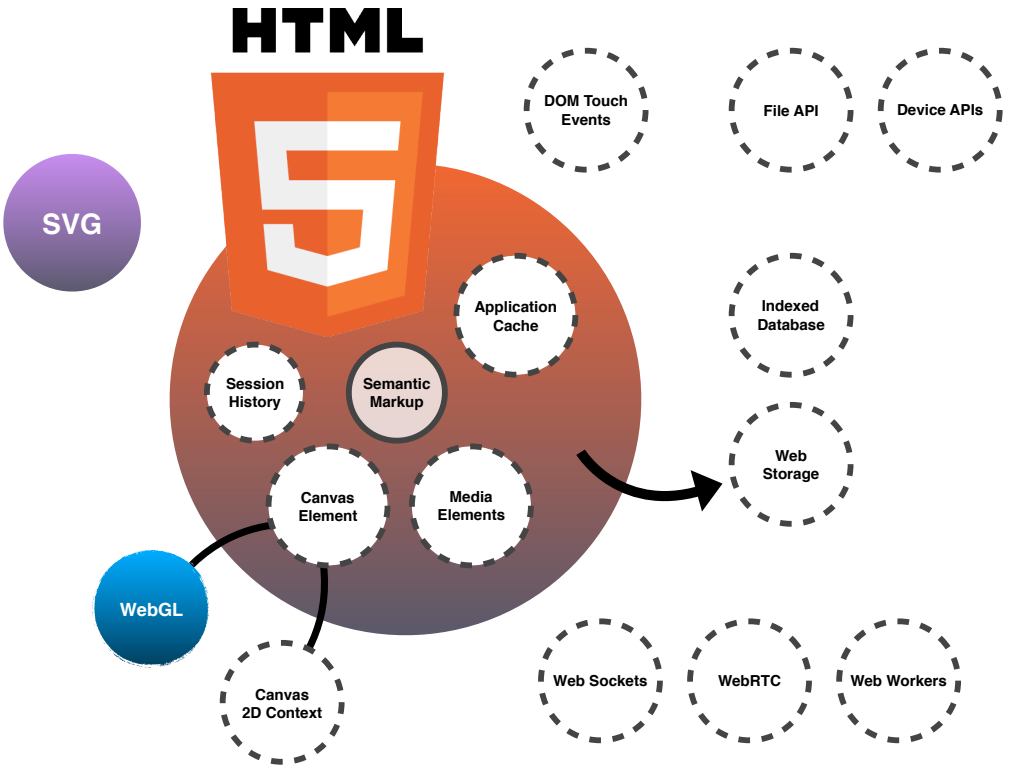
Digitaalisen julkaisemisen kannalta keskeisimmät spesifikaatiot liittyvät sisällön muotoon, joka tässä diplomityössä on määritelty HTML5-taitetuksi. Seuraavassa käsitellään laajan HTML5-spesifikaation keskeinen sisältö digitaalisen julkaisemisen näkökulmasta, jonka jälkeen käydään lävitse CSS3-spesifikaatio ja sen merkitys aiheen kannalta. Molemmat spesifikaatiot ovat tärkeässä osassa esitellen ne teknologiat, joihin tässä diplomityössä toteutetut sovellukset ja niiden avulla tarjottu sisältö perustuvat. Lisäksi käsitellään laaja joukko muita spesifikaatioita, jotka tarjoavat tukevia teknologioita: näihin lukeutuvat muun muassa DOM-kosketustapahtumat ja SVG-vektorigrafiikkaformaatin määrittelevät spesifikaatiot. Osa spesifikaatioista, kuten monet Device API -rajapinnat, ovat vielä keskeneräisiä, mutta mahdollistavat tulevaisuudessa entistä kilpailukykyisempien sovellusten kehityksen ainoastaan selainteknologioita käyttämällä.

3.1.1 HTML5

HTML5 on W3C:n ylläpitämän HTML-spesifikaation viides versio, joka määrittelee verkkosivujen toteuttamiseen tarkoitetun kuvauskielen (Hyper-

Text Markup Language) [12]. Spesifikaation viides versio laajentaa kuvauskieltä, mutta myös poistaa siitä vanhentuneita ja vähän käytettyjä osia - HTML5 määrittelee muun muassa laajemman valikoiman HTML-elementtejä lisäten sisällön kuvailuun käytettävää semantiikkaa [12]. Uuden määrittelyn mukaisesti sisältöä voidaan jäsentää esimerkiksi *article*, *section* tai *aside* -elementeillä [12], joista jokainen jo itsessään määrittelee elementin sisällön suhteen HTML-dokumenttiin. Monien elementtien määrittelyissä käyttökonventio konkretisoidaan painettujen julkaisujen kontekstista, esimerkiksi *aside*-elementti määritellään päätekstistä erotettavaksi kokonaisuudeksi, joka painetussa kirjallisuudessa asetettaisiin tyypillisesti sivun reunaan kainalotekstiksi [12].

Kuvauskielen rikastuminen semanttisesti uuden version myötä tarjoaa siis suoraa hyötyä digitaalisille julkaisuille - sisältöä voidaan jäsentää esimerkiksi maltillisemmalla määrällä HTML-elementeille määriteltäviä luokkia, joka omalta osaltaan selkeyttää itse HTML-syntaksia, mutta myös ulkoasun määrittelemiseen käytettäviä CSS-tyyliresursseja. Uusia elementtejä voidaan hyödyntää myös verkkosivuun pohjautuvan web-applikaation toteutuksessa käyttämällä esimerkiksi *nav*, *header* ja *footer* -elementtejä [12] sovelluksen rakenteen toteuttamisessa. Kuvassa 3.1 on havainnollistettu tässä esiteltäviä ja diplomityössä hyödynnettyjä HTML5-teknologioita. Kuvassa itse HTML5-spesifikaation ytimeen merkittynä semanttinen merkintäkieli.



Kuva 3.1: HTML5:n esitelty ja diplomityössä käytetty konteksti: spesifikaation omat teknologiat on esitetty oranssinharmaalla taustalla W3C:n HTML5-logon[42] yhteydessä, liittyvät teknologiat ja rajapinnat ympärillä. W3C:n rajapinnat on esitetty katkoviivalla, Khronos Groupin WebGL on erotettu sinisellä taustalla, myöhemmin esiteltävä SVG omana merkittävänä teknologianaan violetilla taustalla.

Suuren muutoksen HTML5 toi kuitenkin sivuun sulautettuihin sisältöihin (engl. embedded content), joista tunnetuin on perinteisesti ollut kuvien esittämisen mahdollistava *img*-elementti. Uuden spesifikaation myötä HTML-sivuille on mahdollista lisätä muun muassa ääntä, videota ja tekstityksiä. Uuden mediasisällön esittämisestä vastaavat *audio*, *video*, *source* ja *track* -elementit, joista *source* ja *track* eivät itsessään esitä mitään, vaan niitä käytetään mediaelementtien eli *audion* ja *videon* osana [12]. HTML5:n *track*-elementillä voidaan toteuttaa tekstitykset, kun taas *source*-elementti mahdollistaa useiden vaihtoehtoisten mediaresurssien määrittelemisen mediaelementille. Käytännössä tämä tarkoittaa hyvin usein eri muotoa olevien

[42] `http://w3.org/html/logo/`

resurssien määrittelemistä, sillä spesifikaatio ei ota kantaa esimerkiksi säiliömuotoon (engl. container format) tai pakkausformaattiin (engl. compression format) [12] - HTML5 onkin saanut julkisuutta formaattien välisistä taisteluista. HTML5 paransi myös yleisesti sivulle liitettävän sisällön asemaa standardoimalla muun muassa yleisesti tuetun *embed*-elementin, joka ei kuulunut aikaisempiin HTML4- ja XHTML -spesifikaatioihin [12].

HTML5 toi mukanaan myös piirtoalue-elementin (engl. canvas), joka mahdollistaa rasteroidun grafiikan esittämisen resoluutioriippuvaisella piirtoalueella [12]. Piirtoalueelle esiteltiin myös oma Canvas 2D Context -spesifikaatio, joka mahdollistaa grafiikan esittämisen ja muokkaamisen piirtoalueella ohjelmallisesti JavaScript-rajapinnan avulla [13]. Piirtoalueella voidaan esittää esimerkiksi kuvia, tekstiä, reaaliaikaisesti muuttuvaa grafiikkaa tai sillä voidaan toteuttaa jopa itsenäisiä sovelluksia, kuten pelejä. Piirtoalueen suurimpana rajoituksena on kuitenkin sen resoluutioriippuvaisuus, johon esimerkiksi CSS3 ja SVG voivat tarjota vaihtoehtoisen ratkaisun.

HTML5-spesifikaation osana on esitelty kaksi datan tallentamiseen keskittyvää ratkaisua, joista molemmat ovat omalta osaltaan parantaneet erityisesti web-applikaatioiden toteutusmahdollisuuksia. Näistä ensimmäinen on HTML5-tallennustila (engl. Web Storage), joka ei ole enää osa HTML5-spesifikaatiota, vaan itsenäinen spesifikaatio [14]. Tässä diplomityössä teknologiaan viitataan kuitenkin osana laajempaa teknologista kontekstia, josta käytetään nimitystä HTML5. Tallennustila jakautuu paikalliseen (pysyvään) tallennustilaan (engl. Local Storage) ja sessiokohtaiseen tallennustilaan (engl. Session Storage), joista jälkimmäinen on tarkoitettu sivuston yhden käyttökerran aikaisen datan tallentamiseen [14]. Selain vastaa tallennustilan datan tallentamisesta asiakaslaitteeseen. Paikalliseen eli pysyvään tallennustilaan tallennettu tieto säilyy selaimen ja verkkosivuston käytettävissä kunnes se sivuston tai käyttäjän toimesta poistetaan, esimerkiksi tyhjennettäessä selaimen tietoja. Tietoa tallennetaan tallennustilaan avain-arvo -pareina tekstimuodossa [14], joten tallennustila ei tue suoraan esimerkiksi kuvien tallentamista. Kuvia voidaan kuitenkin säilöä esimerkiksi *data URL* -muodossa base64-enkoodattuna. Käytännössä suurimmaksi rajoitteeksi muodostuvat selaimien implementaatiot, jotka tukevat pääasiassa viiden megatavun tallennusmäärää paikkallisen tallennustilan osalta. [15]

HTML5 määrittelee offline-toiminnallisuuden web-applikaatioille, jonka keskeinen teknologinen osa on HTML5-sovellusvälimuisti (engl. Application Cache) [12]. Spesifikaatio määrittelee manifest-tiedoston, jonka avulla verkkosivu voi listata resurssit, jotka se haluaa selaimen sovellusvälimuistiin [12]. Välimuistitetut resurssit ovat selaimen käytettävissä ilman verkkoyhteyt-

tä, joten esimerkiksi web-applikaatio voi toimia täysin offline-tilassa ensimmäisen latauskerran jälkeen. Sovellusvälimuistia voidaan käyttää myös online-tilassa nopeuttamaan esimerkiksi myöhemmin tarvittavien resurssien käyttöön saamista [16]. Manifest-tiedosto ja sovellusvälimuistin sisältö voidaan päivittää ilman verkkosivun lataamista uudelleen, mikä mahdollistaa niiden dynaamisen käytön - tätä on hyödynnetty muun muassa diplomityössä toteutetussa web-applikaatiossa. Dynaamisesta HTML5-sovellusvälimuistista kerrotaan tarkemmin kohdassa 5.4.2.3.

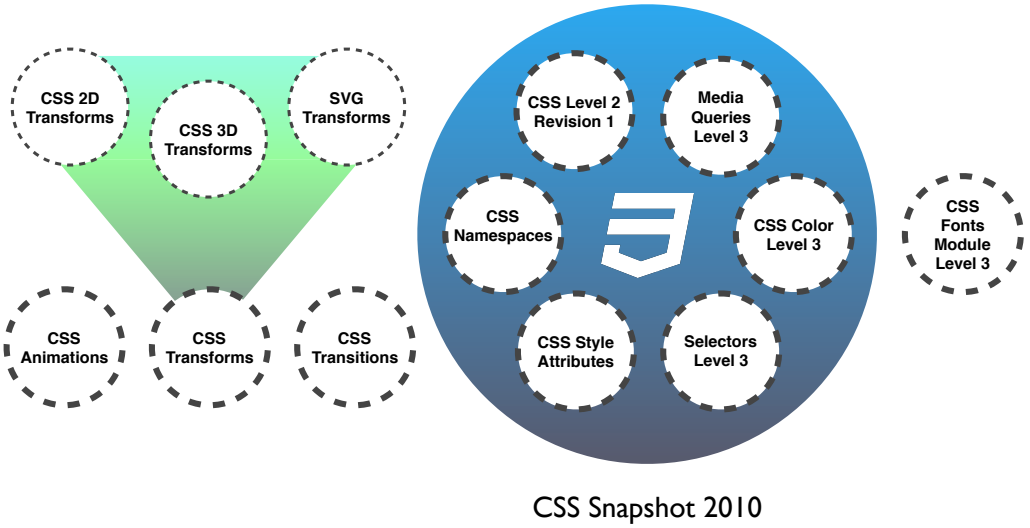
Yhden sivun JavaScript-pohjaista sovellusta toteutettaessa haasteeksi on perinteisesti muodostunut sovelluksen tilojen ja niiden historian tukeminen. HTML5 tarjoaa monipuolisen sessiohistorian hallinnan (engl. session history management), jonka avulla selaimen yhtä verkkosivustoa koskevaa selaushistoriaa voidaan ohjata ohjelmallisesti [12]. Yksisivuinen web-applikaatio voi esimerkiksi pitää kirjaa käyttäjälle esitettävistä näkymistä. Mikäli sovelluksen näkymähierarkiassa on ainoastaan kaksi tasoa, voidaan sessiohistoriaa käyttää helposti takaisin-painikkeen toiminnallisuuden toteuttamiseen.

HTML5 on teknologisena kontekstina huomattavan laaja kokonaisuus ja se tarjoaa joukon teknologioita, joita voidaan käyttää digitaalisessa julkaisemisessa eri jakelukanavien jakelualustojen toteuttamiseen. Se toimii kuitenkin myös ennen kaikkea alustariippumattomana sisältöformaattina, joka helpottaa sisällön tuottamista ja hallintaa. Itse HTML-spesifikaatio kehittyy jatkuvasti ja diplomityön kirjoittamisen loppuvaiheessa siitä onkin alettu valmistella versiota 5.1 [17]. Laajimmillaan HTML5-käsitteellä voidaan tarkoittaa HTML5:n, CSS3:n ja JavaScriptin uusien innovaatioiden synergiaa, jotka yhdessä mahdollistavat natiiveista sovelluksista tutun interaktiotason [18].

3.1.2 CSS3

CSS3 on W3C:n ylläpitämän CSS-tyylikuvauskielen spesifikaation kolmas sukupolvi [19]. Se määrittelee muun muassa uusia valitsimia (engl. selectors) [20], joilla voidaan toteuttaa monimutkaisempia tyylien kohdentamisia HTML-elementeille. Uudet valitsimet eivät kuitenkaan ole CSS3:n suurin muutos, ja monet CSS-kieltä laajentavat kielet, kuten luvussa 3.6.1 käsitelty LESS[43] tarjoavat edistyneempiä mahdollisuuksia tyylien toteuttamiseen juuri valitsimien käsittelyyn liittyvillä menetelmillä. CSS3 on tuonut mukanaan kuitenkin lukuisia muita verkkosivujen visuaaliseen ilmeeseen vaikuttavia teknologioita. CSS3:n keskeiset spesifikaatiot on esitelty kuvassa 3.2.

[43] `http://lesscss.org/`



Kuva 3.2: CSS3:n keskeiset spesifikaatiot. CSS-spesifikaation varsinaiset osat vuoden 2010 tilanteen mukaan on merkitty sinisellä taustalla W3C:n CSS3-logon[44] ympärille. CSS-transformaatiospesifikaation osat visualisoitu vihreälle taustalle.

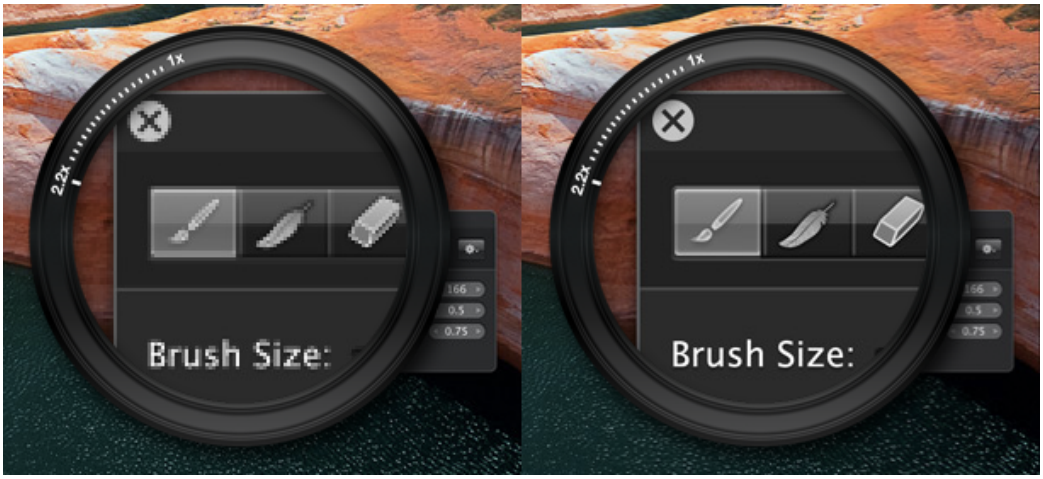
CSS-spesifikaatio koostuu valitsimien ja tyyliattribuuttien (engl. style attributes) määrittelystä, se siis kertoo mitä tyylejä voidaan määritellä ja millä tavalla valituille elementeille. Näiden perusosien lisäksi CSS:n vuoden 2010 snapshot-spesifikaatioon kuuluvat mediakyselyt (engl. Media Queries) [21] ja kolmannen tason CSS-värimoduuli (engl. CSS Color Module Level 3) [22]. CSS3:een liittyy myös läheisesti kolmannen tason CSS-kirjasinmoduuli (engl. CSS Fonts Module Level 3) [23], jota ei ole toistaiseksi liitetty osaksi varsinaista CSS-spesifikaatiota.

Mediakyselyt mahdollistavat tietojen saamisen asiakaslaitteen selaimesta ja näytöstä, mahdollistaen tyylien kohdentamisen etukäteen määritellyillä ehdoilla [21]. Tyypillisimmillään mediakyselyitä käytetään selaimen näköalueen (engl. viewport) leveyden tunnistamiseen, jolloin HTML-sisältö voidaan CSS-tyylien avulla sovittaa leveyteen sopivaksi. Kyse on tällöin responsiivisesta suunnittelusta, josta kerrotaan tarkemmin luvussa 3.2.1. Muita mahdollisia tunnistettavia ominaisuuksia ovat muun muassa orientaatio, kuvasuhde, ja asiakaslaitteen (näytön) mitat [21].

Laitteen pikselitiheys (engl. device pixel ratio) on yleinen mediakyselyehto,

[44] `http://w3.org/html/logo/`

jota ei ole virallisesti spesifikaatiossa. Esimerkiksi Applen Retina-näytöillä varustetuissa laitteissa pikselitiheys on kaksi tarkoittaen pikselimäärän olevan neljäkertainen tavalliseen verrattuna. Varsinainen pikselimäärä näytön fyysisen pinta-alan suhteen vaihtelee eri Retina-laitteiden välillä (esimerkiksi iPhone 5 ja iPad 3), joten pikselitiheys on sidottu näytön simuloimaan resoluutioon. Esimerkiksi Retina-näytöllisen iPadin 2048x1536-resoluutiolle simuloidaan resoluutiota 1024x768, jolloin käyttöliittymäelementeistä tulee samankokoisia kuin 1024x768-resoluutiolla, mutta huomattavasti tarkempia. Mediakyselyitä käytettäessä pikselitiheydeltään suuremmille laitteille voidaan tarjota resoluutioltaan tarkempia kuvia - muussa tapauksessa normaalilaatuiset kuvat voivat vaikuttaa epätarkoilta. Kuvassa 3.3 on havainnollistettu käyttöliittymäelementtien tarkkuuseroa 15” MacBook Pro ja 15” Retina MacBook Pro -laitteiden välillä.



Kuva 3.3: Ote kuvakaappauksesta Applen Retina-näytöllisen MacBook Pro -laitteen tuotesivun Retina-demotyökalusta[45]. Oikealla tarkempi Retina-kuva, käyttöliittymien fyysinen koko on sama molempien laitteiden näytöllä.

CSS-värimoduulin uusi versio tuo yleisesti käytettyjen RGB ja RGB HEX -värien rinnalle RGBA:n, jossa punaisen, vihreän ja sinisen värikanavan lisäksi voidaan säätää alpha-kanavaa eli läpinäkyvyyttä [22]. Käytännössä RGBA mahdollistaa esimerkiksi elementtien taustavärin läpinäkyvyyden, jolloin itse elementin läpinäkyvyyttä ei tarvitse asettaa. Tällöin elementin sisältö, kuten teksti, säilyttää täyden näkyvyyden. Spesifikaatio määrittelee tuen myös HSL ja HSLA -väreille, jotka perustuvat värin määrittämiseen värisävyn, värikyl-

[45] `http://apple.com/macbook-pro/features-retina/`

läisyyden ja kirkkauden avulla [22]. RGBA:n tapaan HSLA:n viimeinen arvo on värin näkyvyys lukujen nolla ja yksi väliltä, yhden tarkoittaessa täyttä näkyvyyttä [22].

Kolmannen tason CSS-kirjasinmoduuli määrittelee uutena teknologiana web-kirjasimet ja niiden käytön [23]. Kirjasinresurssit otetaan käyttöön CSS3:n *@font-face* -säännöllä [23]. Eri selaimet ja alustat käsittelevät uutta sääntöä ja eri kirjasinresursseja vaihtelevilla käytännöillä. Aalto University Magazine -demolehteä varten päädyttiin käyttämään Fontspringin julkaisemaa yleisimmät ongelmatilanteet kattavaa syntaksia[46], joka perustuu Paul Irishin 2009 julkaisemaan alkuperäisversioon[47]. Seuraavassa on esitetty esimerkkisääntö Sentinel-kirjasimen vahvennetulle versiolle. Kirjasimelle määritellään eri selaimien tarvitsemat EOT, WOFF, TTF/OTF ja SVG -resurssit ja sitä voidaan käyttää CSS-tyylimäärittelyissä SentinelBold-nimellä.

```
@font-face {
    font-family: 'SentinelBold';
    src: url('../fonts/sentinel-bold.eot');
    src: url('../fonts/sentinel-bold.eot?#iefix') format('embedded-opentype'),
         url('../fonts/sentinel-bold.woff') format('woff'),
         url('../fonts/sentinel-bold.ttf') format('truetype'),
         url('../fonts/sentinel-bold.svg#SentinelBold') format('svg');
    font-weight: normal;
    font-style: normal;
}
```

Eri alustojen ja selainten web-kirjasintukea on kartoitettu diplomityön kokeellisessa osassa alusta- ja selainkartoituksessa. Spesifikaatio sisältää myös suuren määrän kirjasinten ominaisuuksiin vaikuttavia asetuksia aina kirjasinvälistyksestä ligatuureihin [23], joiden tuki kartoitettiin diplomityön kokeellisessa osuudessa kolmen mobiilikäyttöjärjestelmän oletusselaimille. Kartoitusten tuloksia esitellään luvuissa 6.1.2 ja 6.1.3.

CSS3:n keskeisessä roolissa ovat myös spesifikaatiot, jotka määrittelevät CSS-transformaatiot (engl. CSS Transforms) [24], CSS-siirtymät (engl. CSS Transitions) [25] ja CSS-animaatiot (engl. CSS Animations) [26]. Nämä tekniikat tarjoavat uudenlaisen tavan manipuloida HTML-sisältöä. CSS-transformaatiot määrittelevä spesifikaatio yhdistää 2D- ja 3D-transformaatiot, sekä SVG-transformaation [24]. Monet selainmoottorit, kuten WebKit käyttävät 3D-transformaation implementaatioissaan rautakiihdytystä, eli GPU-pohjaista grafiikan manipulointia [27] erotuksena 2D-transformaation suorittimella suoritettavaan laskentaan. Tämä on vaikuttanut muun muassa diplomityössä toteutetun web-applikaation toteutusratkaisuihin. 3D-transformaatioiden käyttäminen mahdollistaa natiivien sovellusten graafiseen

[46] http://fontspring.com/blog/the-new-bulletproof-font-face-syntax/

[47] http://paulirish.com/2009/bulletproof-font-face-implementation-syntax/

suorituskykyyn vastaamisen web-pohjaisilla sovelluksilla [27]. Monet selainmoottorit käsittelevät myös CSS-siirtymät ja -animaatiot rautapohjaisella kiihdytyksellä [27].

CSS-transformaatiot mahdollistavat elementtien siirtämisen, kääntämisen ja skaalaamisen joko kaksi- tai kolmiulotteisesti riippuen käytetystä transformaatiomuodosta [24]. CSS-siirtymillä elementeille annettuja tyyliarvoja voidaan muuttaa tasaisesti määritellyn ajan suhteen [25]. Haluttu ominaisuus ja siirtymäaika määritellään etukäteen, minkä jälkeen ominaisuuden arvon muuttaminen käynnistää automaattisesti siirtymäajan kestävän siirtymän, jossa arvo muuttuu lineaarisesti lähtöarvosta määriteltyyn kohdearvoon [25]. CSS-siirtymällä voitaisiin esimerkiksi määritellä yhden sekunnin siirtymäaika elementin läpinäkyvyydelle, jolloin arvon muuttaminen animoisi elementin läpinäkyvyyden muutoksen yhden sekunnin aikana. Siirtymille voidaan määritellä myös erilaisia ajoitusfunktioita (engl. timing functions) [25], jotka mahdollistavat esimerkiksi siirtymänopeuden hidastamisen siirtymän loppua kohden. CSS-animaatioissa animaatiot määritellään etukäteen joko käyttäen alku- ja lähtöarvoja tai vaihtoehtoisesti avainkehyksillä (engl. keyframes), jolloin animoitujen ominaisuuksien arvot voivat muuttua avainkehysten mukaisesti useaan kertaan animaation aikana [26]. Animaatiot nimetään ja ne voidaan ottaa käyttöön rajattomalle määrälle elementtejä [26]. Eri transformaatioita ja siirtymän käyttöä on havainnollistettu kuvassa 3.4.

Kuva 3.4: CSS-transformaation 2D-asetuksia, sekä CSS-siirtymällä toteutettu läpinäkyvyyden animointi. Animointi alkaa, kun elementille asetetaan uudeksi läpinäkyvyydeksi arvo *0.2* esimerkiksi automaattisesti *hover*-luokalla tai JavaScriptin avulla. Siirtymän aloituksen viive on nolla sekuntia ja kesto yksi sekunti.

Nämä uudet CSS-tekniikat mahdollistavat elementtien visuaalisen manipuloinnin ilman JavaScript-pohjaisiin animointikirjastoihin tukeutumista. Tämä pienentää JavaScript-resurssien aiheuttamaa kuormaa, yksinkertaistaa selainpohjaisten sovelluksien toimintaa ja tarjoaa mahdollisuuden rautapohjaisen kiihdytyksen käyttöön. CSS3 tarjoaa myös lukuisia uusia tyyliasetuksia, jotka mahdollistavat pikselintarkan visuaalisen toteuttamisen ilman kuvaresursseihin tukeutumista.

3.1.3 Muut spesifikaatiot ja rajapinnat

HTML- ja CSS-spesifikaatioiden lisäksi on joukko muita spesifikaatioita, jotka liittyvät läheisesti HTML5:n kehitykseen ja teknologiseen viitekehyk-

seen. Seuraavassa on käsitelty tässä diplomityössä käytettyjen teknologioiden määrittelyt, sekä muutama web-applikaatioiden kehitykselle keskeinen tulevaisuuden teknologia. Tärkeä spesifikaatio alusta- ja laiteriippumattoman julkaisemisen kannalta on SVG, joka määrittelee kuvauskielen kaksiulotteisen vektorigrafiikan sekä yhdistetyn rasteri- ja vektorigrafiikan esittämiseksi [28]. SVG ja vektoriperusteinen grafiikka mahdollistavat samojen resurssien käytön erikokoisilla ja -resoluutioisilla näytöillä. Rasterigrafiikkaa käytettäessä esimerkiksi käyttöliittymäelementeissä käytetyt kuvaresurssit tulisi optimoida myös pikselitiheydeltään suuremmille näytöille. Käytännössä tämä tarkoittaa vähintään kaksinkertaista resurssimäärää. SVG-resurssit ovat myös itsessään pienempikokoisia, joten ne keventävät verkkosivuston tai -palvelun latausnopeutta.

JavaScriptillä suoritettavalle raskaalle laskennalle ei ole perinteisesti ollut helppoa ratkaisua, vaan kuormantasaus on täytynyt hoitaa manuaalisesti. Tämä on mahdollistanut käyttöliittymän ja selaimen ajautumisen vastaamattomaan tilaan, jossa suoritettava JavaScript vie resursseja enemmän kuin niitä on käytettävissä. Ongelmaa ratkaisemaan on kehitetty taustasuorittajat (engl. Web Workers), jotka mahdollistavat raskaan laskennan suorittamisen erillisessä säikeentapaisessa yksikössä erillään käyttöliittymän koodista [29]. Kehittäjät voivat luoda rajattoman määrän taustasuorittajia, ja kommunikoida niiden kanssa lähettämällä ja vastaanottamalla viestejä JavaScriptistä käsin [29]. Taustasuorittajaimplementaatioiden suorituskykyä on mitattu tämän diplomityön kokeellisessa osiossa, josta kerrotaan tarkemmin luvussa 6.2.

Taustasuorittajat eivät kuitenkaan pääse käsiksi DOM-rakenteeseen, jonka manipuloinnista kosketuksella on tullut käyttäjän ja laitteen välisen interaktion kulmakivi erityisesti tabletti- ja mobiilialustoilla. Jotta kosketukset välittyisivät selaimen kautta HTML-sisällölle, tarvitaan spesifikaatio joka määrittelee DOM-kosketustapahtumat (engl. DOM Touch Events). Tätä varten on kehitetty oma Touch Events -spesifikaatio [30], jota käytännössä kaikki modernit selaimet ja kosketusnäytöllä varustetut laitteet tukevat - diplomityössä toteutetun kartoituksen mukaisesti ainoastaan Windows Phone 7.5 -mobiilikäyttöjärjestelmän IE Mobile -selain ei tue kosketustapahtumia lainkaan ja Windows Phone 8:ssa kosketustapahtumia ei ole toteutettu spesifikaation mukaisesti [15]. Kosketustapahtumat kattavat *touchstart*, *touchend*, *touchmove* ja *touchcancel* -tapahtumat ja ne tukevat useaa samanaikaista kosketusta käyttämällä kosketuksia listaavaa *TouchList*-objektia [30].

AJAX eli Asynchronous JavaScript and XML -tekniikka on ollut pääasiallinen keino yhteydenpitoon palvelimen kanssa selaimessa toimivissa sovelluk-

sissa. Tämän tekniikan rinnalle ovat tulleet web-soketit (engl. Web Sockets), joiden rajapinta määritellään omassa spesifikaatiossaan [31]. Web-soketit mahdollistavat yhteydenpidon palvelimen ja selaimen välillä katkaisematta yhteyttä [31]. Selainten välillä tapahtuvaan viestintään on puolestaan kehitetty WebRTC eli Web Real-Time Communication -tekniikka [32], joka mahdollistaa esimerkiksi jaettujen dokumenttien samanaikaisen muokkaamisen usean käyttäjän toimesta. Muita sovelluksia ovat muun muassa videopuhelut ja tiedostojen jakaminen ilman palvelinta [32].

HTML5:n vanavedessä on esitelty myös WebGL-rajapinta, joka mahdollistaa reaaliaikaisen ja interkatiivisen 3D-grafiikan esittämisen selaimessa [33]. WebGL perustuu OpenGL ES 2.0:aan[48] ja se käyttää HTML5-piirtoaluetta grafiikan renderöintiin [33]. WebGL saa OpenGL:n avulla käyttöönsä rautakiihdytyksen grafiikan ja fysiikan mallinnukseen [33]. Esimerkiksi HTML5-piirtoalueella toimivassa versiossa Angry Birds -pelistä Rovio[49] on tukeutunut WebGL:n käyttöön[50]. Selainten tuki WebGL:lle on yllättävän laajaa myös tabletti- ja mobiilialustoilla, kuten myöhemmin luvussa 6.1.2 esitellyissä alusta- ja selainkartoituksen tuloksissa todetaan.

Tulevaisuuden kannalta merkityksellisiä spesifikaatioita ovat muun muassa tiedostorajapinnan määrittelevä File API [34] ja erinäiset laiterajapinnat (engl. Device APIs). Tiedostorajapinta määrittelee tiedosto-objektin ja sen käsittelyn selainpohjaisille sovelluksille ja mahdollistaa laajemman vuorovaikutuksen asiakaslaitteen tiedostojärjestelmän kanssa [34]. Laiterajapinnat mahdollistavat edistyneemmän tiedon, kuten sijainnin, orientaation, yhteystietojen ja maksutietojen saamisen asiakaslaitteesta, sekä esimerkiksi NFC eli Near Field Communication -tekniikan käyttämisen selainpohjaisella sovelluksella. Monet rajapinnoista ovat vielä keskeneräisiä ja niiden selaintuki on suppeaa: esimerkiksi laitteen orientaation tunnistamiseen käytettävän JavaScript-rajapinnan sijasta kannattaa tällä hetkellä tukeutua hyvin tuettuihin CSS-mediakyselyihin [15, 35]. Eri laiterajapinnat tulevat kuitenkin olemaan keskeisessä roolissa web-applikaatioiden toteutusmahdollisuuksien laajenemisessa ja ne tulevat omalta osaltaan häivyttämään teknologista kuilua web-applikaatioiden ja natiivien sovellusten välillä.

[48] `http://khronos.org/opengles/2_X/`

[49] `http://rovio.com/`

[50] `http://chrome.angrybirds.com/`

3.2 Suunnitteluparadigmat

Tässä luvussa on esitelty diplomityössä käytetyt suunnitteluparadigmat, jotka liittyvät sekä HTML5-muotoisen sisällön toteuttamiseen että selainpohjaisen sovelluksen toimintalogiikan suunnitteluun. Responsiivisen ja adaptiivisen web-suunnittelun osalta painoalue on ollut responsiivisessa suunnittelussa, sillä käytetyt verrattain uudet teknologiat rajaavat omalta osaltaan kohdelaitteiden ja -selainten ryhmää - tarve adaptiiviselle eli mukautuvalle suunnittelulle on siis ollut rajallisempaa. Erityisesti iOS-alustalle toteutetun sovelluksen yhteydessä laitesegmentti on ollut erityisen rajattu: tästä huolimatta esimerkiksi vanhemmat iOS-alustan laitteet asettavat rajoitteita, joihin adaptiivisella suunnittelulla voidaan vastata.

3.2.1 Responsiivinen suunnittelu

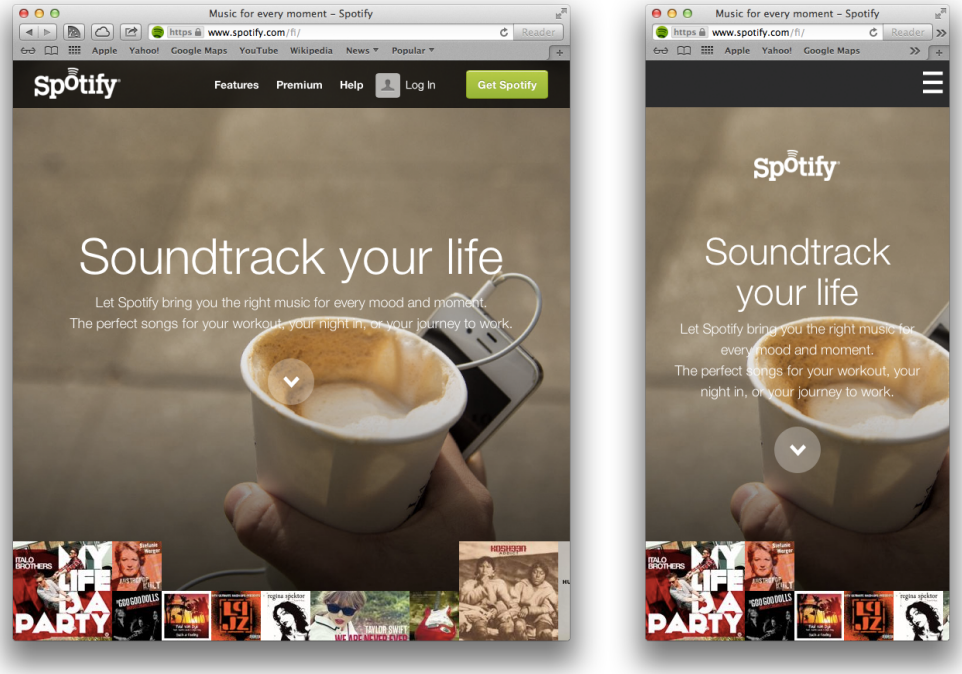
Responsiivisen suunnittelun käsitettä käytti ensimmäisen kerran Ethan Marcotte A List Apart -lehdessä julkaistussa artikkelissaan[51], hän on myös kirjoittanut aiheesta tiiviin teoksen Responsive Web Design [36]. Responsiivisella suunnittelulla pyritään vastaamaan laajaksi kasvaneeseen laitevariaatioon, joka on asettanut kasvavia haasteita web-suunnittelulle. Verkkopalveluiden on tuettava entistä laajempaa laitekirjoa, joka kattaa tabletti- ja mobiililaitteet aina kannettavien tietokoneiden kautta pöytäkoneisiin. Jokaisessa laitesegmentissä mukaan ovat tulleet myös muun muassa suuremman pikselitiheyden omaavat laitteet, mikä omalta osaltaan asettaa uusia vaatimuksia selainpohjaisille käyttöliittymille.

Responsiivisen suunnittelun mukaan web-suunnittelun ja -kehityksen pitäisi ottaa huomioon käyttäjän toiminta ja käyttöympäristö käyttäen hyväksi näytön kokoa, käytettyä alustaa ja laiteorientaatiota [37]. Responsiivisen suunnittelun keinoina ovat mukautuva sivun ulkoasu (engl. flexible layout) ja kuvasisältö sekä CSS-mediakyselyiden älykäs käyttö [37]. Selaimessa toimivan sovelluksen olisi kyettävä mukautumaan käytetyn asiakaslaitteen ominaisuuksien mukaisesti ilman, että suunnittelussa ja kehityksessä jouduttaisiin ottamaan huomioon jokainen markkinoille tullut uusi laite - mikä ei ole edes käytännössä mahdollista [37]. Responsiivisen suunnittelun ydinperiaatteena on tehdä sisällön käyttämisestä mahdollisimman vaivatonta. [36]

Kuvassa 3.5 on esitetty Spotify-musiikkipalvelun verkkosivut, jotka tukeutuvat responsiiviseen ulkoasuun. Ulkoasussa käyttöliittymän eri elemen-

[51] http://alistapart.com/article/responsive-web-design/

tit mukautuvat selaimen leveyteen. Esimerkiksi navigaatiopalkin sisältö siirretään painikkeesta avautuvaan valikkoon esitettäessä sivu pienemmällä leveydellä. Valitussa esimerkissä kuvat eivät mukaudu leveyteen, mitä voidaan hyödyntää ulkoasun suunnittelussa tyylikeinona. Responsiivinen ulkoasu mahdollistaa kuitenkin myös eri resoluutioille sopivampien kuvien tarjoamisen.



Kuva 3.5: Spotify-musiikkipalvelun verkkosivut[52]. Vasemmalla leveämmälle näyttöleveydelle suunniteltu ulkoasu. Ulkoasu mukautuu pienemmille näytöille siirryttäessä, jolloin esimerkiksi navigaatiopalkin sisältö siirtyy painikkeesta avautuvaan valikkoon.

Responsiivisen ulkoasun ja suunnittelun hyödyntämiseen liittyy keskeisesti valinta kiinteälevyisen ulkoasun (engl. fixed layout) ja mukautuvan ulkoasun välillä [38]. Perinteisessä kiinteälevyisessä ulkoasussa sivun sisältö pysyy samanlevyisenä selaimen leveydestä riippumatta, jolloin esimerkiksi mobiiliselaimien tapauksessa näyttöä leveämpi sivu skaalataan oletuksena näytölle sopivaksi. Käyttäjä voi suurentaa sivua esimerkiksi nipistys- tai kaksoisnäpäytys-eleellä. Kiinteälevyistä ulkoasua voidaan kuitenkin käyttää

[52] `http://spotify.com/`

myös yhdessä responsiivisen suunnittelun kanssa, jolloin käytettävää sivun leveyttä vaihdetaan CSS-mediakyselyihin pohjautuen. Nestemäisissä (engl. fluid) ja elastisissa (engl. elastic) ulkoasuissa sivun elementit mukautuvat joko selaimen leveyteen tai käyttäjän kirjasinkokoon jo ilman mediakyselyitä. Mukautuvassa ulkoasussa elementtien leveydet voidaan määritellä prosenttiosuudella isäntäelementistä tai koko sivusta. [39]

3.2.2 Adaptiivinen suunnittelu

Adaptiivisella suunnittelulla viitataan mukautuvan web-suunnittelun toiseen puoleen: progressiiviseen parantamiseen (engl. progressive enhancement). Selainpohjaisia sovelluksia suunniteltaessa olisi otettava huomioon käytettyjä teknologioita tukemattomat laitteet ja rakennettava käyttökokemus alhaalta ylös: edistyneempiä teknologioita käytetään silloin kun ne ovat käytettävissä, muussa tapauksessa tehdään vain perusasiat. Ideana on, että mahdollisimman monet voisivat käyttää palvelua ja ketään ei käytettyjen teknologioiden vuoksi suljettaisi pois palvelusta; sen pitäisi olla siis hyvin saavutettavissa. Progressiivisella parantamisella on myös käänteinen suunnitteluperiaate, sulavalinjainen regressio (engl. graceful degradation), jossa palvelu toteutetaan moderneille selaimille sopivaksi ja teknologioita tukemattomia selaimia tuetaan erityisillä toimenpiteillä. Regressiivisen version käyttökokemuksen ei ole tarkoitus olla miellyttävä, mutta sillä pyritään samalla tavoin saavutettavuuteen kuin progressiivisessakin parantamisessa[53]. [40]

Todellisuus on kuitenkin ideologiaa monimutkaisempi ja käytännössä moderni web-suunnittelu mahdollisimman laajalle laitekannalle tarkoittaa kompromisseja tai ylimääräistä työtä. Adaptiivisen suunnittelun soveltamisessa onkin hyvä tarkastella tehtäviä ratkaisuja pragmaattisesti: kenelle palvelua suunnitellaan ja mitä se tarkoittaa tuettujen laite- ja selainsegmenttien osalta? Adaptiivista suunnittelua voidaan myös soveltaa hyvin monella eri tasolla. Aalto University Magazine -demolehdessä valintojen eteen jouduttiin muun muassa SVG-grafiikan käytön yhteydessä. Kaikki tabletti- ja mobiilialustat eivät tukeneet SVG:tä, minkä takia vaihtoehdoksi toteutettiin PNG-versiot kaikista käyttöliittymässä käytetyistä SVG-resursseista. Raja olisi kuitenkin voitu vetää myös toisin: SVG on web-standardi, jota kaikki modernit selaimet tukevat, se myös helpottaa käyttöliittymän responsiivista suunnittelua merkittävästi, joten teknologian keskeisen roolin vuoksi tuki vanhemmille SVG:tä tukemattomille selaimille olisi voitu pudottaa pois.

Modernien web-teknologioiden kohdalla päätös selaimen tukemisestakaan ei

[53] http://w3.org/wiki/graceful_degredation_versus_progressive_enhancement

välttämättä ole helppo. Esimerkiksi Windows Phone 7.5 -mobiilikäyttöjärjestelmän IE Mobile -selaimesta puuttuvan kosketustapahtumatuen vuoksi kosketuksiin tukeutuvien selainpohjaisten sovelluksien toteuttaminen on käytännössä mahdotonta kyseiselle selaimelle. Yksi valintoja helpottava työkalu on laitesuunnitelma, jossa asiakaslaitteet jaetaan luokkiin tukemiseen kuluvien resurssien ja tuesta saatavien hyötyjen, kuten tulojen mukaisesti. Suunnitelman mukaisesti alemmissa luokissa tuen toteuttaminen on kallista ja vastaavasti tuesta saatava hyöty pienempää. Mallia voidaan käyttää rajan vetämiseen selain- tai laitetuen laajuuden suhteen. [41]

Adaptiivisen suunnittelun kantavana tekijänä ovat laitteen, selaimen ja ominaisuuksien tunnistaminen (engl. feature detection). Suunniteltu verkkosivu tai sovellus voi mukautua asiakaslaitteesta ja -selaimesta tunnistettujen tietojen perusteella. Sovellus voi esimerkiksi tunnistaa tukeeko käytetty selain web-kirjasimia ja käyttää niitä CSS-tyylimäärittelyissä tuen perusteella. Ominaisuuksien tunnistamiseen on olemassa lukuisia valmiita ohjelmistokirjastoja, joista yksi on luvussa 3.5.2 kuvattu JavaScript-kirjasto Modernizr.

Toinen esimerkki progressiivisen parantamisen käytöstä diplomityössä liittyy toteutetun web-applikaation dynaamiseen HTML5-sovellusvälimuistiin, jota käytetään ainoastaan selaimissa, jotka tukevat HTML5-sovellusvälimuistia. Sovellusvälimuistin tuki tunnistetaan käyttämällä Modernizr-kirjastoa ja sovellus toimii myös ilman tukea. Dynaaminen HTML5-sovellusvälimuisti on siis progressiivinen parannus, joka ehostaa toteutettua sovellusta.

3.2.3 Yhden sivun sovellukset

HTML suunniteltiin alunperin käyttökonventiolle, jossa verkkosivuston käyttäjät liikkuvat eri HTML-resurssien välillä. Yhden HTML-sivun lataaminen on kuitenkin raskas operaatio ja hyvin usein pienen informaatiomäärän noutamisen vuoksi turhaa. Tämän vuoksi kehitettiin AJAX eli Asynchronous JavaScript And XML, jolla voitiin suorittaa pyyntöjä palvelimelle ohjelmallisesti verkkosivun sisältä lataamatta sivua uudelleen. AJAX on mahdollistanut yhden sivun web-sovellukset, joissa sovellus rakentuu ainoastaan yhden HTML-sivun varaan [42]. Sovellus toimii JavaScriptin avulla ja on vuorovaikutuksessa palvelimen kanssa tarvittaessa, esimerkiksi käyttäjän toimesta. [43]

Yhden sivun sovelluksen hyöty käyttäjälle syntyy kahden maailman yhdistymisestä: toisaalta SPA eli Single Page Application tarjoaa työpöytäsovelluksen välittömyyden, mutta se on myös saavutettavissa kuten verkkosivusto. Mikowski ja Powell kuvaavat kirjassaan [43] kuinka JavaScriptiä voidaan hyö-

dyntää sovelluksen kaikilla tasoilla aina käyttöliittymästä taustapalvelimelle asti - käytettyihin ohjelmistoihin kuuluvat jQueryn lisäksi muun muassa Node.js[54], Socket.IO[55] ja MongoDB[56], joiden avulla tietoa voidaan siirtää JSON-muodossa. Mikowskin ja Powellin mukaan JavaScript ja sille läheiset teknologiat, kuten HTML5 ja CSS3 ovat vähitellen kehittyneet riittävän hyviksi, jotta ne ovat voineet haastaa perinteisesti suositut SPA-ympäristöt, kuten Flashin ja Javan. Selaimessa toimiva JavaScript on myös maailman levinnein suoritusympäristö mahdollistaen sovellusten suorittamisen mobiililaitteista pöytäkoneisiin. Asiakaslaitteiden suorituskyvyn kasvaessa laskentaa on mahdollista siirtää entistä enemmän palvelimelta asiakaslaitteelle. [43]

Web-applikaatioiden kehittämisessä yhden sivun sovelluksista on tullut vakiintunut suunnitteluparadigma, joka mahdollistaa natiivien sovellusten käyttökokemuksen tavoittelemisen. Monet JavaScript-pohjaiset ohjelmistokehykset, kuten luvussa 3.4.7 käsitellyt Ember.js ja Backbone.js toteuttavat MVC-mallin[57], jota voidaan käyttää yhden sivun sovelluksen toteuttamisessa. MVC:llä viitataan perinteisesti taustapalvelimella toteutettuun jakoon mallin (engl. model), näkymän (engl. view) ja ohjaimen (engl. controller) välillä, millä pyritään tekemään selkeä jako datan, esitystason ja toimintalogiikan välille [44]. MVC voidaan toteuttaa käyttöliittymässä osana SPA:ta [44]. Esimerkiksi HTML5-sovellusvälimuistia hyödyntämällä voidaan yhden sivun sovelluksesta tehdä täysin itsenäinen, ensimmäisen latauskerran jälkeen palvelimesta riippumaton sovellus.

JavaScriptillä toteutettujen yhden sivun sovelluksia voidaan käyttää myös osana natiiveja sovelluksia tai natiivi sovellus voi tukeutua yhden sivun sovellukseen täysin, jolloin voidaan ajatella kyseessä olevan wrapper-sovellus. Natiiveja sovelluksia voidaan toteuttaa myös käyttämällä web-teknologioita, joita käännetään mahdollisuuksien mukaan esimerkiksi natiiveiksi käyttöliittymäelementeiksi. Myöhemmin luvussa 3.4.8 käsitellään Sencha Touch ja PhoneGap, jotka mahdollistavat kyseisen toteutustavan.

3.3 Suorituskykyohjeistot

Web-applikaation suunnitteluun liittyy keskeisesti alustariippumattomuus ja sovelluksen optimointi eri alustoille ja laitteistoille. Selaimella käytettäville ja verkossa toimiville sovelluksille löytyy useita suorituskykyohjeistoja, joita

[54] http://nodejs.org/

[55] http://socket.io/

[56] http://mongodb.org/

[57] http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

kehittäjät voivat noudattaa. Vuonna 2012 julkaistussa diplomityössä [45] on käsitelty suorituskykyisen ja alustariippumattoman sovelluksen toteutusta HTML5-tekniikalla mobiilialustoille. Diplomityössä hyödynnetään Soudersin suorituskykyohjenuoria [46], jotka ovat käyttökelpoisia myös tässä diplomityössä toteutetussa web-applikaatiossa. Steve Souders työskentelee Googlella suorituskyvystä vastaavana pääinsinöörinä ja on kerännyt suorituskykyohjeistonsa työskennellessään Yahoo!:ssa vastaavassa työtehtävässä[58]. Seuraavassa käydään lävitse ne Soudersin kirjassa *High Performance Web Sites* esitetyt suorituskykyohjenuorat, jotka ovat olleet keskeisiä Stage Framework -ohjelmistokehyksen ja Aalto University Magazine -demolehden web-applikaation toteutuksessa.

Tee vähemmän HTTP-pyyntöjä

HTTP-pyyntöjen vähentäminen on keskeisin web-suunnittelussa tehtävä suorituskykytoimenpide. Eri resurssien lataaminen erillisillä HTTP-pyynnöillä aiheuttaa pyyntöjen raskaan kehystämisen vuoksi turhaa latauskuormaa, mikä hidastaa palvelun toimintaa. HTTP-pyyntöjä voidaan vähentää esimerkiksi yhdistämällä resursseja, kuten kuvia tai JavaScript-tiedostoja. Myös web-soketit (engl. Web Sockets) ja paikalliset tietokantaratkaisut, kuten Indexed Database -rajapinta ja HTML5-tallenustila (engl. Web Storage) mahdollistavat HTTP-pyyntöjen vähentämisen [16]. [46]

Gzip-pakkaa resurssit

Web-palvelimen tulisi mahdollisuuksien mukaan pakata kaikki lähetettävät resurssit, sillä se on yksinkertaisin keino pienentää lähetettävän datan määrää. Gzip-pakkauksella pakatut resurssit pienenevät yleensä 70%, Steve Soudersin esimerkissä eri resursseilla saavutettu pakkaustaso oli 56-73%. [46]

Laita CSS-tyyliresurssit alkuun

Tyyliresurssien sijoittaminen HTML-dokumentin alkuun saa ne selaimen käytettäväksi aikaisessa vaiheessa, mikä mahdollistaa visuaalisen palautteen antamisen käyttäjälle mahdollisimman nopeasti [46]. Erittäin kriittiset tyylit, kuten esimerkiksi musta sivun taustaväri voidaan sijoittaa inline-tyyliksi HTML-dokumenttiin.

Laita komentosarjat loppuun

Komentosarjat eli pääasiallisesti JavaScript-resurssit estävät rinnakkaisen resurssien lataamisen ja HTML-dokumentin renderöinnin sijainnistaan eteenpäin, joten ne tulisi sijoittaa kaikkien muiden resurssien jälkeen - ideaalisti

[58] `http://stevesouders.com/`

HTML-dokumentin loppuun. [46]

Ulkoista JavaScript- ja CSS-resurssit

JavaScript- ja CSS-resurssien ulkoistaminen selkeyttää HTML-dokumenttia ja tekee eri resurssien ylläpitämisestä ja päivittämisestä vaivattomampaa [46]. Suorituskykyvaikutus syntyy selaimen mahdollisuudesta välimuistittaa ulkoiset resurssit, joita käytetään useammassa HTML-dokumentissa: inline-tyylit ja komentosarjat ovat osa vaihtuvasisältöisiä HTML-dokumentteja, joten ne joudutaan lataamaan aina uudelleen [46]. Tilanne on hieman erilainen esimerkiksi yhden sivun sovelluksissa, mutta vaikutus resurssien ylläpitoon ja hallittavuuteen on kuitenkin sama.

Kutista JavaScript-resurssit

JavaScript-koodia voidaan kutistaa erilaisin toimenpitein säilyttämällä kuitenkin sen toiminta samana [46]. Esimerkiksi muuttujanimien korvaaminen lyhemmillä sekä kommenttien ja tyhjän tilan poistaminen pienentävät siirrettävän datan määrää [46]. JavaScript-resurssien kutistamiseen on olemassa lukuisia työkaluja, kuten Google Closure Compiler[59].

Mahdollista AJAX-kutsujen välimuistitus

Riippuen palvelun luonteesta AJAX-kutsut sisältävät paljon dataa, joka pysyy samana eri pyyntökertojen välillä. Tästä syystä myös AJAXilla tehtävien kutsujen välimuistittaminen olisi hyvä mahdollistaa. [46]

Steve Souders paneutuu syvemmin myöhemmin julkaistussa kirjassaan *Even Faster Web Sites* muun muassa JavaScriptin, XHR-kutsujen ja kuvien optimointiin [47]. Kirjassa käsitellään myös responsiivisen web-applikaation suunnittelua, tällä kertaa eri merkityksellä: miten esimerkiksi taustasuorittajia (engl. Web Workers) voidaan hyödyntää, jotta sovelluksen käyttöliittymä pysyisi käytettävissä suuren kuorman alla [47]. Seuraavassa on poimittu diplomityön kannalta merkittävimmät suorituskyvyn parantamiseen tähtäävät tekniikat.

[59] `http://developers.google.com/closure/compiler/`

Ensimmäisen kuorman jakaminen

Tyypillisesti kaikkia HTML-dokumentin mukana ladattavia resursseja ei tarvita välittömästi tai edes tyypillisen käyttökerran aikana. Sovelluksen ensimmäistä käynnistymistä voidaan nopeuttaa jakamalla kuormaa myöhemmin ladattavaksi. [47]

Komentosarjojen lataaminen ilman estämistä

Komentosarjojen eli skriptien lataaminen estää tyypillisesti selainta lataamasta muita resursseja samanaikaisesti. Komentosarjojen rinnakkainen lataaminen toisten komentosarjojen ja muiden resurssien kanssa on kuitenkin mahdollista esimerkiksi käyttämällä useampaa domain-nimeä. [47]

Asynkronisten komentosarjojen yhdistäminen

Mikäli komentosarjoja ladataan rinnan, on mahdollista, että niiden suoritusjärjestys sekoittuu. Toisistaan riippuvat komentosarjat voidaan yhdistää samaksi resurssiksi oikeassa suoritusjärjestyksessä. [47]

Tehokkaan JavaScriptin kirjoittaminen

JavaScript-koodin tehostamiseen on olemassa lukuisia käytäntöjä. Steve Souders pureutuu kirjassaan muun muassa DOM-rakenteen käsittelyyn ja raskaan laskennan aiheuttamaan problematiikkaan. [47]

Kuvien optimoiminen

Souders esittää useita keinoja kuvien optimoimiseen formaattitasolla, mutta tarjoaa vaihtoehdoksi myös muun muassa kuvien yhdistämisen suuremmiksi sprite-kuviksi eli yhden kuvan kuvakokoelmiksi [47]. Sprite-kuvat on kuitenkin haastettu myöhemmin tässä luvussa Paul Irishin parhaiden käytäntöjen listassa CSS3:n tuomien mahdollisuuksien valossa.

CSS-valitsimien yksinkertaistaminen

CSS-valitsimien välillä on merkittäviä suorituskykyeroja. Perusesimerkkinä ovat *id* ja *class* -attribuuteilla tehtävät valinnat, joissa *id*:llä elementti saadaan paikannettua välittömästi, kun taas pelkällä luokka-valitsimella joudutaan käymään koko DOM-rakenne lävitse. CSS-valitsimien tarkalla harkinnalla voidaan parantaa CSS-tyyliresurssien suorituskykyä. [47]

Steve Soudersin perinteisten suorituskykyyn vaikuttavien ohjenuorien lisäksi on olemassa joukko toimenpiteitä, joita voidaan soveltaa erityisesti modernien web-applikaatioiden suorituskyvyn parantamiseksi. Näistä yhden parhaiden käytäntöjen listan [16] on kerännyt Paul Irish, joka on ol-

lut mukana kehittämässä muun muassa jQuery ja Modernizr -kirjastoja[60]. Irishin parhaat käytännöt on esitetty seuraavassa.

Käytä HTML5-tallennustilaa evästeiden tilalta

Modernien web-sovellusten tulisi tallentaa mahdollisimman vähän tietoa evästeisiin, sillä niiden sisältö liitetään mukaan jokaiseen HTTP-pyyntöön, mukaan lukien esimerkiksi yhden sivun JavaScript-sovelluksesta tehtyihin XHR-kutsuihin. Evästeiden koon kasvaessa myös jokaisen HTTP-kutsun koko kasvaa. HTML5-tallennustila (engl. Web Storage) mahdollistaa evästeisiin tallennettavien tietojen tallentamisen pysyvään Local Storage ja sessiokohtaiseen Session Storage -tallennustilaan. [16]

Käytä CSS-siirtymiä JavaScript-animaatioiden tilalta

CSS-siirtymät mahdollistavat useimpien ominaisuuksien, kuten tekstin varjostuksen, sijainnin, taustan ja värin animoimisen. Siirtymä voidaan käynnistää esimerkiksi lisäämällä CSS-asetuksen arvoa muuttava HTML-luokka elementille. CSS-siirtymät tarvitsevat huomattavasti vähemmän koodia verrattuna JavaScript-animointikirjastoihin, mikä vähentää selaimelle lähetettävän datan määrää. Siirtymät käyttävät myös GPU-pohjaista kiihdytystä, joten ne ovat mahdollisimman sulavia. [16]

Käytä asiakaspuolen tietokantoja palvelinpyyntöjen tilalta

Asiakaspuolen tietokannat, kuten Indexed Database -rajapinta, tarjoavat mahdollisuuden HTTP-pyyntöjen vähentämiselle. Esimerkiksi suuren sähköpostilaatikon sisältö voidaan tallentaa ennemmin paikallisesti, kuin pyytää jatkuvasti uudelleen palvelimelta. Myös HTML5-tallennustilaa (engl. Web Storage) voidaan käyttää joissain tapauksissa - esimerkiksi lomakkeen täytön etenemisen tallentaminen on todettu nopeammaksi tallennustilaa käyttämällä. [16]

JavaScript-parannukset tuovat huomattavia suorituskykyetuja

JavaScript-kieli ja sen implementaatiot kehittyvät siinä missä muutkin web-teknologiat. Esimerkiksi JavaScriptin 1.6-versiossa laajennettiin *Array*-prototyyppiä uusilla metodeilla, jotka mahdollistavat monimutkaisempia operaatioita taulukon tiedoilla, mutta perinteistä *for*-silmukkaa nopeammin. Muita esimerkkejä ovat muun muassa *JSON.parse* ja *String.trim* -metodit. Kun kieleen tehtävät parannukset implementoidaan kaikkiin selaimiin, voivat ne tuoda huomattavia suorituskykyetuja. [16]

[60] `http://paulirish.com/about/`

Käytä HTML5-sovellusvälimuistia myös live-sivustoille, ei ainoastaan offline-sovelluksille

HTML5-sovellusvälimuistia (engl. Application Cache) ja cache.manifest-tiedostoa voidaan käyttää verkkosivustoilla myös muutenkin kuin offline-toiminallisuuden saavuttamiseksi. Verkkosivuston perusrakenne voidaan ajatella muuttumattomaksi, johon tuodaan aina uutta sisältöä esimerkiksi JSON-muodossa. Nämä muuttumattomat staattiset resurssit voidaan välimuistittaa sovellusvälimuistiin, jota selaimet voivat optimoida tehokkaasti. [16]

Ota käyttöön rautapohjainen kiihdytys parantaaksesi visuaalista kokemusta

Selaimet pystyvät hyödyntämään GPU-pohjaista kiihdytystä, joka tekee dynaamisista visuaalisista operaatioista huomattavasti sulavampia. Animoitu siirto, kääntö, skaalaus ja läpinäkyvyyden muuttaminen hyötyvät kaikki rautapohjaisesta kiihdytyksestä. Operaatiot voidaan toteuttaa suoraan GPU:lla ilman animoitavan tason sisällön uudelleenpiirtoa. [16]

Taustasuorittajat vastaavat laskennallisesti vaativiin operaatioihin

Taustasuorittajilla (engl. Web Workers) on kaksi huomattavaa etua: ne ovat nopeita, ja kun ne suorittavat niille annettua tehtävää, käyttöliittymä pysyy käytettävissä. Taustasuorittajia voidaan käyttää esimerkiksi pitkän dokumentin tekstin formatointiin, syntaksin korostamiseen tai kuvien ja suurien taulukoiden käsittelyyn. [16]

HTML5-lomakeattribuutit ja -syöttötyypit

HTML5 tuo lukuisia parannuksia lomakkeisiin: uudet syöttötyypit tarjoavat selaimen omat implementaatiot esimerkiksi värin tai päivämäärän valitsemiseksi. *Placeholder*-attribuutti mahdollistaa ohjetekstien automaattisen näyttämisen tyhjässä kentässä ja *autofocus* mahdollistaa kentän valitsemisen sivun latautuessa. *Required*-attribuutti estää selainta lähettämästä lomaketta mikäli kenttä on täyttämätön ja *pattern*-attribuutti mahdollistaa kentän sisällön testaamisen säännöllisiä lausekkeita vastaan. [16]

Käytä CSS3-efektejä raskaiden yhden kuvan kuvakokoelmien (engl. image sprite) tilalta

CSS3 tarjoaa useita ominaisuuksia visuaalisesti tarkkojen suunnitelmien toteuttamiseksi. Sprite-kuvatiedoston vaihtuminen 100 kilotavuun CSS-tyylejä on suuri parannus ja poistaa jälleen yhden HTTP-pyynnön. Mainitsemisen arvoisia ominaisuuksia ovat muun muassa lineaariset ja ympyrägradientit, kulmien pyöristykset, elementtien varjostus, RGBA, transformaatiot ja CSS-

maskit. [16]

Web-soketit tarjoavat nopeamman ja pienikaistaisemman vaihtoehdon XHR:lle

Web-soketit käyttävät vähemmän kaistaa kuin esimerkiksi XHR-kutsut, sillä niiden paketit on kehystetty kevyemmin. Suurin hyöty saadaan kuitenkin huomattavasti nopeammasta vasteajasta, jonka johdosta web-soketit soveltuvat paremmin reaaliaikaista tiedonsiirtoa vaativiin sovelluksiin. [16]

Lisäyksenä Paul Irishin CSS3:een liittyviin suosituksiin voidaan mainita myös pienikokoisten rasterikuvien lisääminen CSS-tyylien sekaan *data URL* -muodossa. Menetelmä vähentää omalta osaltaan HTTP-kutsuja, vaikka kuvan esittäminen tässä muodossa viekin hieman enemmän bittejä. Lisäksi Irishin käytännössä mainittuihin JavaScript-parannuksiin liittyy myös toinen näkökulma: esimerkiksi *JSON.parse*-metodi on huomattavissa määrin parantanut JSON-formaatin tietoturvaa JavaScriptillä käsiteltäessä [48].

3.4 Ohjelmistokehykset

Tässä alaluvussa käydään lävitse HTML5-julkaisemiselle keskeisiä ohjelmistokehyksiä. Kaikkia esiteltyjä ohjelmistokehyksiä ei ole käytetty diplomityössä, mutta niiden hyödyntäminen olisi ollut mahdollista muissa toteutusratkaisuissa, sekä osana Stage Framework -ohjelmistokehyksellä julkaistavaa sisältöä. Ohjelmistokehykseen päätyvät julkaisijat voivat siis hyödyntää esimerkiksi Laker compendiumin resursseja omissa julkaisuissaan.

3.4.1 Laker compendium

Laker compendium on kokoelma resursseja, jotka auttavat suunnittelemaan ja toteuttamaan digitaalisia julkaisuja HTML5-tekniikalla[61]. Resurssit koostuvat muun muassa digitaalisille julkaisuille sopivasta HTML5-sivupohjasta ja siihen liittyvistä CSS-tyyleistä. Resurssit on pyritty rakentamaan mahdollisimman geneerisiksi, jotta ne sopisivat mahdollisimman monelle julkaisulle. Lakerin resurssit on suunniteltu itsenäisiä julkaisuja silmälläpitäen, joten esimerkiksi web-palvelin ei ole välttämättömyys julkaisun levittämiseen[61].

Laker tarjoaa muun muassa responsiivisen grid-asettelun ja ohjelmistokirjaston tekstin tavuttamiseen tavutuksen sallivien HTML-luokkien avulla. Lakerin tarjoamia resursseja voidaan käyttää itse toteutetun sovelluksen osana

[61] `http://lakercompendium.com/`

tai ohjelmistokehyksen osaksi kehitetyn iOS-sovelluksen avulla, joka on paketoitu osaksi omaa ohjelmistokehystä, Baker eBook Frameworkiä.

3.4.2 Baker eBook Framework

Baker eBook Framework sisältyy Laker compendiumiin, mutta on itsenäinen ohjelmistokehys HTML5-taitetun sisällön julkaisemiseksi iOS:lle wrapper-sovelluksena[62]. Ohjelmistokehys ottaa HTML5-taitetun sisällön sisäänsä erillisinä HTML-sivuina, joista jokainen vastaa yhtä julkaisun sivua. Baker käyttää julkaisun kokoamiseen JSON-syntaksin manifest-tiedostoa, joka noudattaa HPub eli HTML Publication -tiedostoformaattia[63]. Formaattia ei pidä sekoittaa EPUB-formaattiin[64], jota käytetään yleisesti digitaalisten julkaisujen kuvailemiseen ja toteuttamiseen, muun muassa Sigil[65] ja eCub[66] -sovellusten avulla.

Baker toteuttaa iOS-sovelluksen, joka on toteutettu Applen vaatimusten mukaisesti Objective-C:llä ja iOS SDK:lla. Sovellus perustuu yhteen Scroll View -näkymään, joka sisältää useita Web View -näkymiä. Käyttäjä pystyy pyyhkäisemään Scroll View -näkymää ja siirtymään Web View -näkymien välillä, jotka sisältävät julkaisun sisällön eli HTML-taitetut sivut HPub-manifestin määrittelemässä järjestyksessä. Ohjelmistokehyksen käyttäjä pystyy muokkaamaan iOS-sovellustaan vapaasti, esimerkiksi määrittelemällä haluamansa aloituskuvat ja sovellusikonit. Valmiin sovelluksen levittämiseen App Storessa vaaditaan jäsenyys maksullisessa iOS Developer Program -ohjelmassa[67].

Baker-ohjelmistokehyksessä on toteutettu kaksi renderöintitapaa sivujen välillä siirtymiselle: kuvakaappauksia käyttävä ja three-cards -renderöinti. Renderöintitapa voidaan valita manifest-tiedostossa muiden asetuksien, kuten latausanimaatioiden ja -tekstien visuaalisen ilmeen lisäksi[68]. Kuvakaappauksiin pohjautuvassa renderöinnissä käyttäjälle esitetään uudelle sivulle saavuttaessa kuvakaappaus sivusta ennen kuin sivu on ehditty ladata kokonaan. Three-cards -renderöinnissä sivuja pidetään välimuistissa kolme kerrallaan: esimerkiksi nykyinen ja kaksi seuraavaa, riippuen edellisen pyyhkäisyn eli sivunvaihdon suunnasta. Eri renderöintimenetelmillä pyritään

[62] http://bakerframework.com/
[63] http://github.com/Simbul/baker/wiki/hpub-specification/
[64] http://idpf.org/epub/
[65] http://code.google.com/p/sigil/
[66] http://juliansmart.com/ecub/
[67] http://developer.apple.com/programs/ios/
[68] http://bakerframework.com/tutorials/30/manifest/

nopeuttamaan visuaalisen palautten esittämistä käyttäjälle.

3.4.3 Friar eBook Framework

Friar eBook Framework on ohjelmistokehys HTML5-taitettujen digitaalisten julkaisujen tuottamiseksi Android-alustalle[69]. Friar on hyvin samankaltainen Bakerin kanssa ja se käyttääkin HPub-manifestia julkaisun määrittelemiseen. Friar sisältää Android-sovelluksen, joka ottaa sisäänsä HTML5-taitetun sisällön yhdessä HPub-manifestin kanssa. Käyttäjä voi muokata sovellusta vapaasti ja julkaista sen Google Play -sovelluskaupassa. Julkaisua varten vaaditaan Google Play Developer Console -tunnus[70], jonka rekisteröinti on maksullinen iOS Developer Program -ohjelman tapaan.

3.4.4 HTML5 Boilerplate

HTML5 Boilerplate tarjoaa pohjan HTML5-käyttöliittymälle tarjoamalla keskeiset resurssit valmiina pakettina, josta kehittäjät voivat karsia ylimääräiset osat pois[71]. Boilerplate sisältää muun muassa mobiilialustoille tärkeitä meta-tageja, web-applikaation sovellusikonisääntöjä, CSS-tyylien alustuksia, jQuery- ja Modernizr-kirjastot, sekä web-palvelimen (Apache) suorituskykyyn vaikuttavia asetuksia[71]. HTML5 Boilerplate on itsenäinen pohja web-pohjaisille sovelluksille tai verkkosivustoille, jonka tarkoituksena on nopeuttaa kehitystyön aloittamista.

3.4.5 Bootstrap

Bootstrap[72] on käyttöliittymäohjelmistokehys web-pohjaisille käyttöliittymille. Se tarjoaa erinomaisen pohjan responsiiviselle ulkoasulle ja sisältää oman grid-asettelun. Bootstrap tarjoaa kattavat CSS-tyylikirjastot muun muassa tekstisisällön, taulukoiden, lomakkeiden, painikkeiden, navigaatiopalkkien, leivänmurupolkujen, kuvien ja latausanimaatioiden tyylittämiseen. Se sisältää myös Glyphicons -ikonikirjaston[73], jota voi käyttää eri käyttöliittymäelementtien kanssa. Bootstrapin mukana tulee myös joukko JavaScript-

[69] http://friarframework.com/
[70] http://play.google.com/apps/publish/
[71] http://html5boilerplate.com/
[72] http://twitter.github.com/bootstrap/
[73] http://glyphicons.com/

kirjastoja, jotka mahdollistavat muun muassa kuvakarusellien, modaalinäkymien, välilehtien ja vierityksen seurannan toteuttamisen.[74]

3.4.6 LESS Framework ja ZURB Foundation

LESS Framework[75] on grid-järjestelmä, jota on käytetty muun muassa Laker compendiumissa ja Baker eBook Framework -ohjelmistokehyksessä. LESS Framework sisältää neljä eri asettelua ja kolme typografia-asetusta, joita voidaan käyttää ulkoasun suunnittelun apuna. Asettelut on jaettu mobiiliin, leveään mobiiliin, tablettiin ja oletusasetteluun. Asetteluiden leveys vaihtelee 320:n ja 992:n pikselin välillä. Konsistenttien asetteluiden käyttäminen toteutuksissa helpottaa sisällön suunnittelua ja uudelleenkäyttöä eri asetteluiden välillä. Vastaavanlainen käyttöliittymien asetteluiden toteuttamiseen tarkoitettu ohjelmistokehys on ZURB Foundation[76], joka tarjoaa muun muassa kahdeksan erilaista valmiiksi suunniteltua asettelupohjaa eri tyylisille palveluille.

3.4.7 Ember.js ja Backbone.js

Ember.js[77] ja Backbone.js[78] ovat molemmat JavaScriptiin pohjautuvia ohjelmistokehyksiä, jotka tarjoavat MVC-malliin pohjautuvan valmiin frontendin web-pohjaiselle sovellukselle. Ohjelmistokehykset pyrkivät siirtämään datan ja sen käsittelyn pois DOM-rakenteesta, jolloin DOM pysyy ainoastaan datan ilmentymänä. Backbone.js on hieman kevyempi kuin Ember.js, mikä toisaalta tekee Emberistä varteenotettavamman vaihtoehdon täysiverisille JavaScript-sovelluksille. Backbone.js soveltuu keveytensä ansiosta paremmin pienille sovelluksille ja sovelluksien osatehtäville. Ohjelmistokehykset siirtävät myös tapahtumien käsittelyn datan sisältäville JavaScript-objekteille: Ember.js pystyy muun muassa päivittämään reaaliaikaisesti kaikissa näkymissä näytettävät datan instanssit, kun data päivittyy sovelluksessa.[79]

[74] http://twitter.github.com/bootstrap/

[75] http://lessframework.com/

[76] http://foundation.zurb.com/

[77] http://emberjs.com/

[78] http://backbonejs.org/

[79] http://net.tutsplus.com/tutorials/javascript-ajax/game-on-backbone-and-ember/

3.4.8 Sencha Touch ja PhoneGap

Sencha Touch on mobiilisovelluksille suunniteltu ohjelmistokehys, joka mahdollistaa natiivinnäköisten web-pohjaisten sovellusten toteuttamisen HTML5-tekniikalla[80]. Se tarjoaa käyttöliittymäelementtikirjaston ja laajan JavaScript-kirjaston eri käyttöliittymätoiminnallisuuksien toteuttamiseen.

PhoneGap on ohjelmistokehys, joka mahdollistaa natiivien sovellusten toteuttamisen HTML, CSS ja JavaScript -teknologioilla. PhoneGap tukee käytännössä kaikkia mobiilikäyttöjärjestelmiä ja tarjoaa yhtenäisen rajapinnan natiivien ominaisuuksien käyttämiseksi[81]. PhoneGap toteuttaa wrapper-sovelluksen eri mobiilikäyttöjärjestelmille ja tarjoaa helpon keinon julkaista web-teknologioilla toteutetun sovelluksen natiivina eri alustoille.

Sencha Touch ja PhoneGap ovat usein käytetty yhdistelmä ja ne tarjoavat SPA ja MVC -suunnitteluperiaatteet toteutettuna web-teknologioilla, mutta käännettynä natiivin sovelluksen muotoon.

3.5 Ohjelmistokirjastot

Tässä alaluvussa käsitellään diplomityössä käytettyjä ohjelmistokirjastoja. Kirjastot ovat siis pääasiassa selaimessa toimivia käyttöliittymäkirjastoja, ja ne on toteutettu JavaScriptillä. Raja ohjelmistokirjaston, -kehyksen ja tukevan teknologian välillä on joskus häilyvä, esimerkiksi myöhemmin tukevissa teknologioissa esitelty SQLite voidaan luokitella myös ohjelmistokirjastoksi. Tässä esitetty jako palvelee kuitenkin tätä työtä koskevaa teknologista viitekehystä.

3.5.1 jQuery

jQuery[82] on noussut käytännössä Internetin de facto JavaScript-kirjastoksi ja sen markkinaosuudeksi arvioidaan jopa yli 50% kaikista verkkosivustoista[83]. Sen kantavina teknologisina ansioina ovat DOM-rakenteessa liikkumisen ja manipuloinnin, DOM-tapahtumakäsittelyn ja AJAX-kutsujen yksinkertaistaminen. Keskeisinä periaatteina ovat myös JavaScript-koodin selkeyttäminen ja erityisesti jQueryn alkuaikoina alustariippumattoman ra-

[80] http://sencha.com/products/touch/
[81] http://phonegap.com/
[82] http://jquery.com/
[83] http://w3techs.com/technologies/overview/javascript_library/all/

japinnan tarjoaminen JavaScript-toteutuksiltaan vaihtelevilla selainmarkkinoilla. jQuery on helposti laajennettavissa modulaarisesti, joten sille löytyy paljon JavaScript-kirjastoja niin kutsuttuina plugineina. jQuerya käytetään tässä diplomityössä kaikissa JavaScript-resursseissa.

3.5.2 Modernizr

Modernizr on selainten ominaisuuksia tunnistava JavaScript-kirjasto, joka tarjoaa kattavan rajapinnan selaimen tukemien ominaisuuksien kartoittamiseksi käyttöliittymätasolla[84]. Se tukee erinomaisesti uusia HTML5:n ja CSS3:n ominaisuuksia, joten sitä käytettiin sekä suoritetussa alusta- ja selainkartoituksessa että toteutetun web-applikaation osana. Modernizr tarjoaa kehittäjille helpon tavan tunnistaa voidaanko tiettyä ominaisuutta käyttää käyttäjän selaimessa. Tällaisia ominaisuuksia voivat olla esimerkiksi HTML5-sovellusvälimuisti, CSS3-animaatiot, SVG-tuki tai web-kirjasinten eri formaatit.

Ominaisuuden selaintuen tunnistaminen voidaan toteuttaa käyttäen neljää eri keinoa riippuen tarkistettavasta ominaisuudesta: tarkistamalla ominaisuuden läsnäolo globaalilta objektilta, kuten *window* tai *navigator*; luomalla elementti ja tarkistamalla ominaisuuden läsnäolo luodulta elementiltä; luomalla elementti ja tarkistamalla metodin läsnäolo ja kutsumalla metodia; luomalla elementti tai muuttamalla ominaisuuden arvoa ja tarkistamalla onko ominaisuus pitänyt arvon. Monimutkaisempien ominaisuuksien tunnistamisen kannattaa kuitenkin suosiolla jättää Modernizrin tapaisille tunnistuskirjastoille. [49]

3.5.3 PhotoSwipe

PhotoSwipe on laajaa mobiilialustakirjoa tukeva JavaScript-kirjasto, joka tarjoaa yksinkertaisen ja yleisimpiä käyttöliittymäeleitä tukevan kuvagallerian[85]. PhotoSwipe mahdollistaa kuvien selaamisen kuvagalleriassa käyttämällä pyyhkäisyä ja niiden koon muuttamisen nipistämällä. Se tukee myös muun muassa automaattista kuvaesitystä ja kuvatekstejä. PhotoSwipe-kirjastoa käytettäessä määritellään kuvagallerian aktivoivat elementit, sekä siihen sisällytettävien kuvien lähteet ja kuvatekstit (esimerkiksi DOM-rakenteesta). Aalto University Magazine -demolehdessä PhotoSwipe-kirjastoa käytettiin tarjoamaan kuvagalleria jokaisen artikkelin kuville. Vaik-

[84] `http://modernizr.com/`
[85] `http://photoswipe.com/`

ka itse artikkelin kuvien koot määriteltiin responsiivisen suunnittelun periaatteen mukaisesti eri näytöille sopivaksi, tarjottiin kuvagalleriassa aina täysikokoiset versiot kuvista, joita käyttäjät pystyivät edelleen suurentamaan.

3.5.4 Swipe.js

Swipe.js on Brad Birdsallin kehittämä JavaScript-kirjasto, joka implementoi pyyhkäisyeleen käyttäen CSS3:n 3D-transformaatiota[86]. Swipe.js on julkaistu GPL ja MIT -lisensseillä ja sitä on käytetty Stage Frameworkin osana implementoitaessa lehtien selausnäkymää. Erityisen käyttökelpoisen toteutetun selausnäkymän kannalta Swipe.js-kirjastosta teki resistanttien reuna-aluiden toteutus, jota on hyödynnetty myöhemmin myös selausnäkymän vierityksen toteutuksessa CSS3:lla.

3.6 Tukevat teknologiat

Tässä alaluvussa käsitellään diplomityön aikana käytettyjä tukevia teknologioita, jotka eivät ole ohjelmistokehyksiä tai -kirjastoja. Tällaiset teknologiat ovat tyypillisimpiä HTML5-muotoista sisältöä tuotettaessa. Kaksi mainitsemisen arvoista web-kehitystä tukevaa teknologiaa, joita tässä diplomityössä ei aktiivisesti käytetty ovat CoffeeScript-kieli ja Google Closure Compiler -palvelu. Molemmat näistä tähtäävät suorituskykyisempiin JavaScript-resursseihin, jotka ovat tärkeä osa web-kehitystä. CoffeeScript on kevyt kieli, joka kääntyy JavaScriptiksi ja pyrkii huomaamattomasti parantamaan käytettyjä ohjelmointiparadigmoja - kieli on yhteensopiva kaikkien JavaScript-kirjastojen, kuten jQueryn kanssa ja lopputuloksena syntyy nopeammin suoritettavaa koodia[87]. Google Closure Compiler on työkalu valmiiden JavaScript-resurssien tiivistämiseen ennen jakelua käyttäjille. Tiivistämisen lisäksi työkalu pyrkii kuitenkin myös nopeuttamaan koodia muilla menetelmillä, kuten analysoimalla sen rakennetta[88].

[86] http://swipejs.com/

[87] http://coffeescript.org/

[88] http://developers.google.com/closure/compiler/

3.6.1 LESS

LESS on dynaaminen tyylikuvaskieli, joka laajentaa CSS-tyylikuvauskielen ominaisuuksia[89]. LESSissä kehittäjät voivat asettaa usein käytetyt arvot, kuten värikoodit muuttujiin, jolloin kyseistä värisävyä voidaan hallita keskitetysti muuttamalla ainoastaan yhtä arvoa. LESS mahdollistaa myös muun muassa CSS-valitsinten (engl. selectors) sisäkkäisen käytön, joka selkeyttää tyylimäärittelyitä ja mahdollistaa elementtien välisen hierarkian esittämisen sisennyksillä. LESSissä valitsimia ja muuttujia voi käsitellä myös funktioina, joille voidaan antaa parametreja samaan aikaan niitä käytettäessä. Myös laskuoperaatiot numeraalisia arvoja sisältävillä muttujilla on mahdollista[89].

LESS-tyylit voidaan kääntää CSS-tyyleiksi eri vaiheissa riippuen käytetystä tuotantotekniikasta - valmiit resurssit voidaan kääntää ennen tuotantoympäristöön siirtämistä, palvelimella (mahdollisesti käyttäjän pyytäessä resurssia), tai vasta selaimessa. Aalto University Magazine -demolehden tapauksessa LESS-tyylit käännettiin Ruby[90]-skriptillä tuotettaessa yhtä lehden numeroa, mutta esimerkiksi Stage Framework -ohjelmistokehyksen web-applikaatiossa LESSiä käytettiin JavaScriptin avulla asettamaan ladatun lehden tyylit uuteen kontekstiin web-applikaation DOM-rakenteen sisällä.

LESS ei kuitenkaan ole ainoa vaihtoehto, vaan tarjolla on muitakin. Yksi suosituimmista on Compass[91], joka on CSS-resurssien tuottamiseen tarkoitettu ohjelmistokehys, joka perustuu Sass-tyylikuvauskieleen[92], joka puolestaan LESSin tapaan laajentaa CSS-tyylikuvauskieltä.

3.6.2 Haml

Haml on HTML:n tapaan kuvauskieli, joka kuvaa HTML-muotoista sisältöä käyttämättä kuitenkaan HTML:ää[93]. Haml-kielen avulla voidaan toteuttaa uudelleen käytettäviä HTML-pohjia yksittäisistä elementeistä aina kokonaisiin sivuihin asti. Haml-tulkkien implementaatioita löytyy useimmille ohjelmointikielille, mutta virallinen ja alkuperäinen on olemassa ainoastaan Ruby-kielelle, jota käytettiin demolehden sisällön käsittelemiseen ja tuottamiseen tässä diplomityössä. Haml nopeuttaa ja yksinkertaistaa HTML-pohjien luo-

[89] http://lesscss.org/
[90] http://ruby-lang.org/
[91] http://compass-style.org/
[92] http://sass-lang.com/
[93] http://haml.info/

mista verrattuna esimerkiksi Rybyn omaan ERB-luokkaan.

Aalto University Magazine -demolehden tapauksessa Haml-pohjia käytettiin yksinkertaisimmillaan generoitaessa jokaiselle HTML-sivulle *head*-tagin sisältöä, mutta monimutkaisimmillaan tuotettaessa lehden artikkeleiden sisältöä ja sisältösivun elementtejä. Haml-pohjia kutsuttiin Ruby-skriptistä antaen samalla muuttujia arvoina määritellyille parametereille. Haml-pohjassa esimerkiksi yhden sisältösivun elementin luomisessa voitiin testata onko pohjaa kutsuttu kuvalla vai ilman, eli tarvitseeko sisältösivun nostolle luoda paikka kuvalle kyseiselle artikkelille vai ei. Itse kuva voitiin liittää generoituun elementtiin määritellyn *id*-attribuutin ja LESSin avulla.

3.6.3 SQLite

Diplomityössä käytettiin SQLite-tietokantamoottoria[94] toteutettaessa dynaamista HTML5-sovellusvälimuistia, josta kerrotaan tarkemmin kohdassa 5.4.2.3. SQLite on kevyt, palvelimeton SQL-tietokanta, jota voidaan käyttää web-palvelimella esimerkiksi PHP[95]:n avulla. SQLite on suosittu tietokantamoottori web-ympäristön lisäksi mobiilikäyttöjärjestelmissä, joissa niitä käytetään muun muassa natiivien sovellusten tietojen tallentamiseen. Dynaamisen HTML5-sovellusvälimuistin tapauksessa käytettiin yhdestä tiedostosta koostuvaa SQLite-tietokantaa tallentamaan viimeksi välimuistitettuja resursseja. Käyttäjille generoitiin laitteeseen ja selaimeen kytketty uniikki tunniste, joka tallennettiin asiakaspuolella HTML5-tallennustilaan tai keksiin (engl. cookie), josta se pystyttiin lukemaan suoraan selaimen pyytäessä HTML5-sovellusvälimuistin manifest-tiedostoa. Web-applikaation toteutuksessa tämä olisi mahdollistanut esimerkiksi offline-tilaan ladattavien lehden numeroiden muistamisen käyttäjän tunnisteen perusteella.

[94] `http://sqlite.org/`
[95] `http://php.net/`

Luku 4

Referenssijulkaisut

Tässä luvussa käsitellään muita olemassa olevia digitaalisia julkaisuja, jotka ovat joutuneet valitsemaan omat ratkaisunsa digitaaliselle julkaisemiselle tabletti- ja mobiilialustoilla. Osassa referenssijulkaisuista on tukeuduttu yksinomaan HTML5-teknologiaan, mikä tekee niistä erityisen mielenkiintoisia tämän diplomityön kannalta. Referenssijulkaisujen valinnassa on kuitenkin teknisen toteutuksen mielenkiintoisuuden lisäksi painotettu julkaisujen merkitystä globaalisti tai kotimaisilla markkinoilla.

4.1 Financial Times

Financial Times on yksi maailman suurimmista pääasiallisesti talouteen keskittyvistä julkaisuista[96]. Lehteä julkaiseva The Financial Times Ltd on julkaissut muun muassa taustoittavan How To Spend It Magazine -julkaisun natiivina sovelluksena Applen App Storessa, mutta tukeutuu päälehden julkaisemisessa lehden verkkoversion lisäksi web-applikaatioon, joka on suunniteltu yksinomaan iOS-alustoille [50]. Lehdestä on toteutettu natiivit hybridisovellukset niin Android- kuin Windows 8 -alustoillekin, mikä omalta osaltaan viittaa uusien HTML5- ja CSS3-teknologioiden parempaan tämänhetkiseen suorituskykyyn nimenomaan iOS-alustoilla. Natiivit hybridisovellukset ovat ladattavissa Google Play -kaupasta ja Windows Storesta[97].

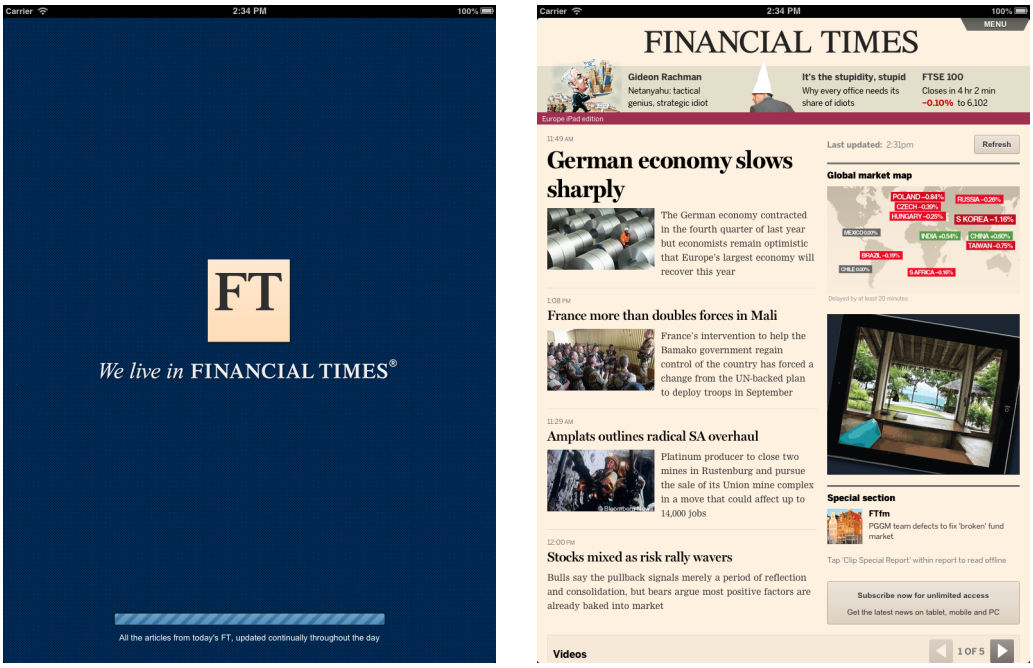
Tukeutuminen web-applikaatioon on perusteltua, sillä se on voittanut muun muassa GSMA Global Awardsin vuoden 2012 palkinnon Best Mobile Innovation for Publishing päätyen esimerkiksi muissa kategorioissa palkittujen

[96] `http://aboutus.ft.com/`

[97] `http://apps.ft.com/`

Applen, Samsungin ja Rovion seuraan[98].

Visuaalisesti web-applikaatio noudattaa muille laitteille julkaistujen sovellusten kanssa lähes identtisesti Financial Timesin verkkojulkaisua ja sisältö onkin toteutettu HTML5-pohjaisesti sovelluksesta riippumatta [50]. Tämä omalta osaltaan helpottaa julkaisua useammalle eri sovellukselle, sillä uuden sisällön tuottamisessa voidaan tukeutua samaan formaattiin. Sovelluksissa julkaistu sisältö on maksullista, johon käyttäjät pääsevät käsiksi tilaamalla lehden digitaaliset sisällöt, sisältäen myös lehden verkkosivuston ja muiden alustojen sovellukset. Vahva digitaalinen läsnäolo siivitti Financial Timesin digitaalisten tilausten määrän ohittamaan lehden painetun version tilaukset ensimmäistä kertaa vuoden 2012 keväällä[99]. Financial Timesin web-applikaatio on esitetty kuvissa 4.1 ja 4.2.



Kuva 4.1: Financial Times -lehden web-applikaation käynnistysnäkymä ja aloitussivu. [50]

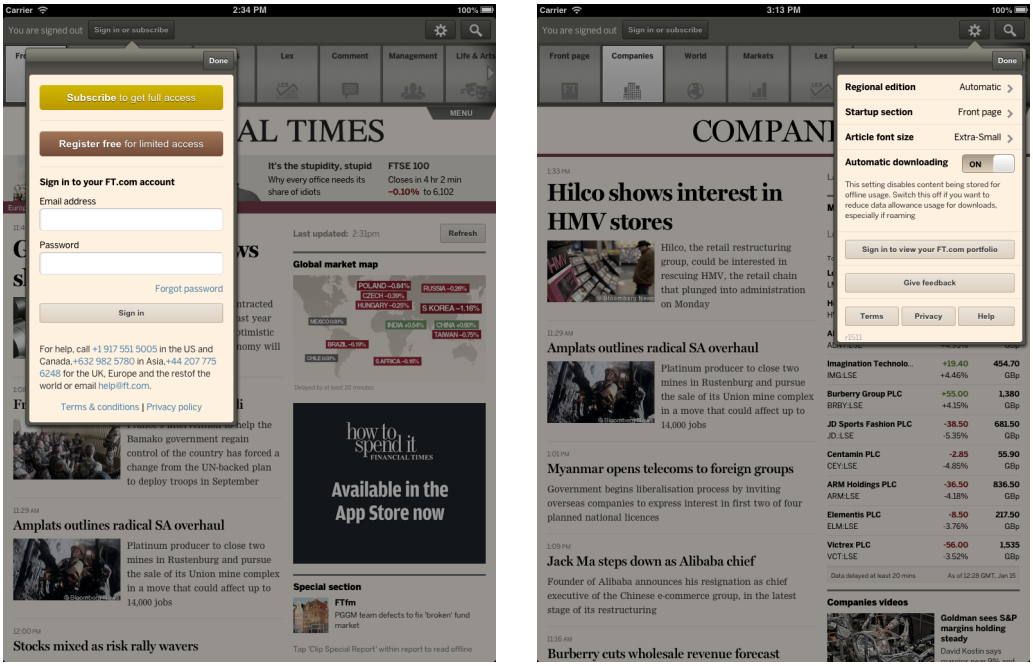
Tekniseltä toteutukseltaan Financial Timesin lehden selausnäkymä perustuu CSS3:n 3D-transformaatioon. Yhden sivun sovelluksessa lehden sivu-

[98] http://globalmobileawards.com/awards-history/winners-2012/

[99] http://aboutus.ft.com/2012/07/27/ft-digital-subscriptions-surpass-print-circulation-globally/

ja on ladattuna DOM-rakenteeseen kerrallaan kolme: nykyisen sivun lisäksi edellinen ja seuraava sivu, mikä mahdollistaa selaussuunnan vaihtamisen kesken sivunvaihdon. Käyttäjä siirtyy sivulta toiselle käyttäen natiiveista sovelluksista tuttua elettä, pyyhkäisyä, jota voidaan hyödyntää DOM-kosketustapahtumien avulla. Financial Timesin ratkaisussa pyyhkäisyn aikana muutetaan kahden näkyvillä olevan sivun sijaintia (vaaka-akselilla) erillisinä DOM-päivityksinä 3D-transformaation avulla. Sivujen isäntäelementiin ei siis kosketa 3D-transformaatiolla. [50]

Web-applikaation sivujen selaaminen pystysuunnassa on niin ikään toteutettu samalla CSS3-tekniikalla. Financial Times on toteuttanut oman version iOS-alustoilta tutusta kineettisestä vierityksestä, jossa sivun liike hidastuu kosketuksen loputtua. Etenkin pyyhkäisyyn käytetty animointialgoritmi eroaa kuitenkin iOS-alustoilla käytetystä. Selattavassa sivussa ei myöskään ole toteutettu kimpoamista (engl. bounce back), jossa vierityksen on mahdollista ylittää sivun reuna ja kimmota takaisin asettaen sivun reuna kiinni selaimen näkymän (engl. viewport) reunaan. Sovelluksen ja sen sisällön ulkoasussa on käytetty responsiivista suunnittelua, sillä sama sovellus mukautuu iPadin ja iPhonen näytöille. [50]



Kuva 4.2: Financial Times -lehden web-applikaation navigaatiopalkin tunnistautumis- ja asetusvalikot. Modaalinäkymän tapaan toimivat valikot animoivat taustan tummemmaksi CSS3:n avustuksella. [50]

Web-applikaatiossa julkaistun lehden viimeisin sisältö ladataan automaattisesti offline-lukemista varten iOS-laitteen Safari-selaimen HTML5-sovellusvälimuistiin (engl. HTML5 Application Cache). Financial Timesin mukaan yhden lehden koko sisältö mahtuu suunnilleen 25-50 megatavun tilaan[100]. Tämä tarkoittaa lehden pyytävän lisätilaa käyttäjältä, erityisesti käyttäjän käyttäessä iOS 6 -versiota aiempaa ohjelmistoversiota - iOS 6:sta lähtien HTML5-sovellusvälimuistin koko on oletuksena ollut 25 megatavua [15]. Perinteisen sisällön lisäksi web-applikaatio tarjoaa videoita, joita ei tallenneta HTML5-sovellusvälimuistiin. Käyttäjä siis tarvitsee toimivan verkkoyhteyden videoiden katsomista varten[100].

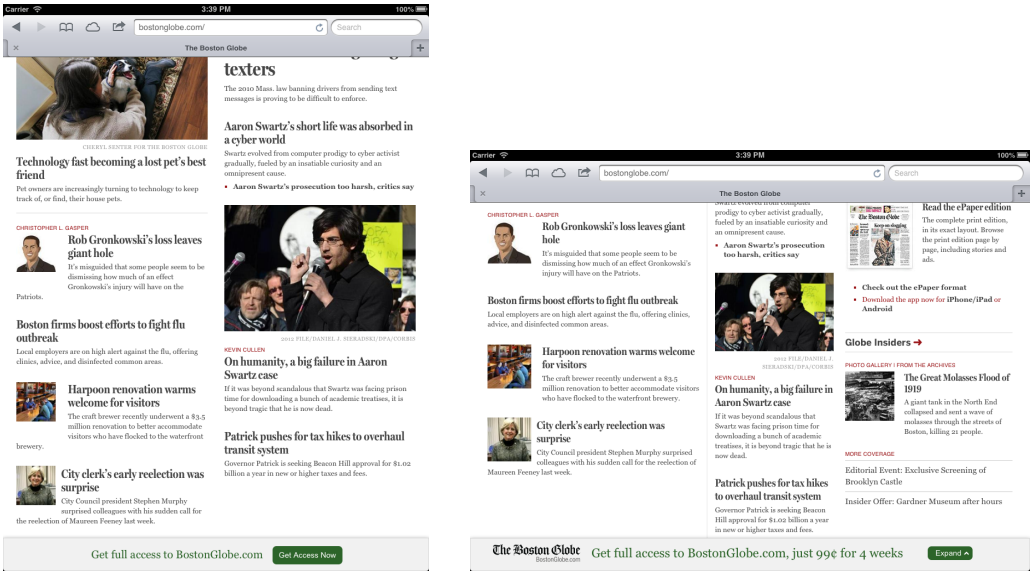
Käyttäjä voi mukauttaa web-applikaatiota valitsemalla oletukseksi maantieteellisen sijainnin mukaan toimitetun version, ensimmäisenä aukeavan osion, sekä käytetyn kirjasinkoon. Käyttäjä voi myös poistaa käytöstä sisällön automaattisen lataamisen [50]. Financial Timesin web-applikaatiosta vastaa-

[100] `http://apps.ft.com/ftwebapp/faq.html`

vat kehittävät julkaisevat kirjoituksia ja sovelluksissaan käyttämiään koodeja avoimilla lisensseillä sivustolla labs.ft.com.

4.2 Boston Globe

Boston Globen [51] lähestymistapa digitaaliseen julkaisemiseen eroaa Financial Times -lehdestä. Lehdestä on saatavilla ePaper-versiot eli näköislehdet natiiveina sovelluksina Applen App Storesta iOS-alustoille ja Google Play -kaupasta Android-laitteille. Lehden reaaliaikaisesti pävittyvä sisältö on kuitenkin tarjolla eri laitteille optimoituna yhden osoitteen alla, joka yhdistää Boston Globen verkkosivut, mobiilisivut ja web-applikaation. Lehti ei ole panostanut juurikaan web-applikaation hiomiseen tietyille alustoille esimerkiksi natiiveilla eleillä, vaan keskittynyt responsiivisen suunnittelun tarjoamiin etuihin. Lehti on aidosti alustariippumaton ja mukautuu jo esimerkiksi yhden laitteen orientaatiomuunnoksiin, kuten seuraavassa kuvassa 4.3 on esitetty.



Kuva 4.3: Boston Globe -lehden verkkosivu mukautuu kaksipalstaisesta kolmipalstaiseen orientaation vaihtuessa vaakasuuntaiseksi Applen iPad-laitteella. Lisäksi uudella palstalla näytetään nostoja, joita ei näytetty kaksipalstaisessa ulkoasussa. [51]

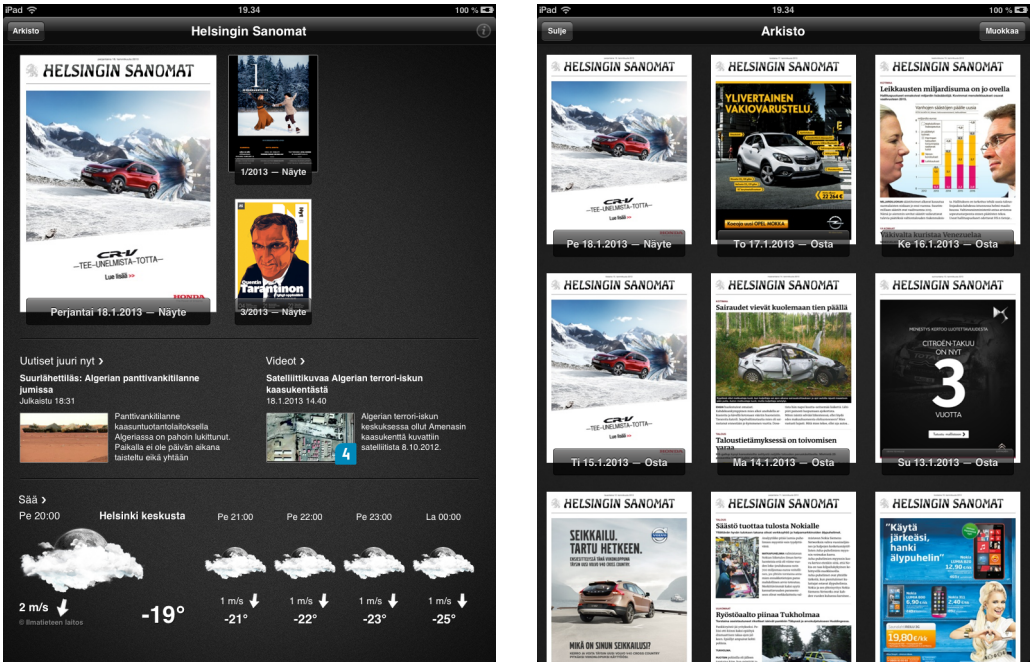
Lehti ei myöskään perustu yhden sivun sovellus -paradigmaan, vaan käyttäjä liikkuu eri artikkelisivujen välillä käyttäen tavallista selaimen navigaatiota. Boston Globe tarjoaa digitaalista sisältöään viikottaisella hinnalla tarjoten ensimmäiset neljä viikkoa erityisen edulliseen hintaan. Lehden digitaalista sisältöä tilaamaton käyttäjä näkee lyhyet koosteet uutisista ja artikkeleista, mutta siirtyy automaattisesti tilaussivulle määritellyn ajan jälkeen[101].

4.3 Helsingin Sanomat

Helsingin Sanomat on seurannut verkkosivustonsa kanssa The New York Timesin esimerkkiä. Käyttäjät voivat lukea verkkosivuston uutisia ainoastaan rajoitetun määrän, jonka jälkeen he kohtaavat maksumuurin (engl. paywall). Käyttäjät voivat lukea kaikkia uutisia ja artikkeleita kuukausittaista maksua vastaan. Molemmissa palveluissa verkkosivuston etusivu ja uutiskategorioiden pääsivut ovat kuitenkin selattavissa vapaasti, joka mahdollistaa mainostilan myymisen vanhaan tapaan palvelusta maksamattomien kävijöiden nähtäväksi. Verkkosivusto ei itsessään ole responsiivinen, joten se näyttää samalta laitteella kuin laitteella. Sivusto kuitenkin tunnistaa selkeästi pienemmän resoluution ja tarjoaa vaihtoehtoista mobiilisivua[102].

Helsingin Sanomat tarjoaa verkkosivustonsa lisäksi hybridisovelluksia iOS, Android, Windows Phone ja Symbian -alustoille. Sovellukset ovat maksuttomia, mutta sisältö on saatavilla samalla digitaalisen sisällön tilauksella kuin verkkosivustonkin. Käyttäjille tarjotaan myös mahdollisuus ostaa irtonumeroita. Hybridisovelluksien kautta julkaistut lehdet sisältävät pääosin samat uutiset ja artikkelit kuin painettukin lehti, mutta kyseessä ei silti ole näköislehti. Lehti on taitettu käytetylle laitteelle sopivaksi käyttäen pohjana HTML5-sisältöä. Mainokset ovat dynaamisia, joten ne eivät vastaa painetun lehden mainoksia ja muuttuvat lehteä selattaessa. Kuvassa 4.4 on esitetty Helsingin Sanomien iOS-sovelluksen aloitus- ja arkistonäkymät. Aloitusnäkymässä näytetään uusin päälehti, sekä viikottain ja kuukausittain ilmestyvien liitelehtien uusimmat numerot. Lisäksi näkymässä on nostettu yksi uutisartikkeli ja yksi samaan Sanoma[103]-konserniin kuuluvan Nelonen[104]-televisiokanavan toimittama uutisvideo. Näkymän alaosassa käyttäjälle esitetään myös seuraavien tuntien säätiedot, joiden päivittämiseen sovellus myös pyytää lupaa käyttää laitteen sijaintitietoja. [52]

[101] http://bostonglobe.com/
[102] http://hs.fi/
[103] http://sanoma.fi/
[104] http://nelonen.fi/



Kuva 4.4: Helsingin Sanomien iOS-sovelluksen aloitusnäkymä ja arkisto. Käyttäjät voivat kirjautua palveluun ja käyttää aktiivista digitilausta tai ostaa irtonumeroita. [52]

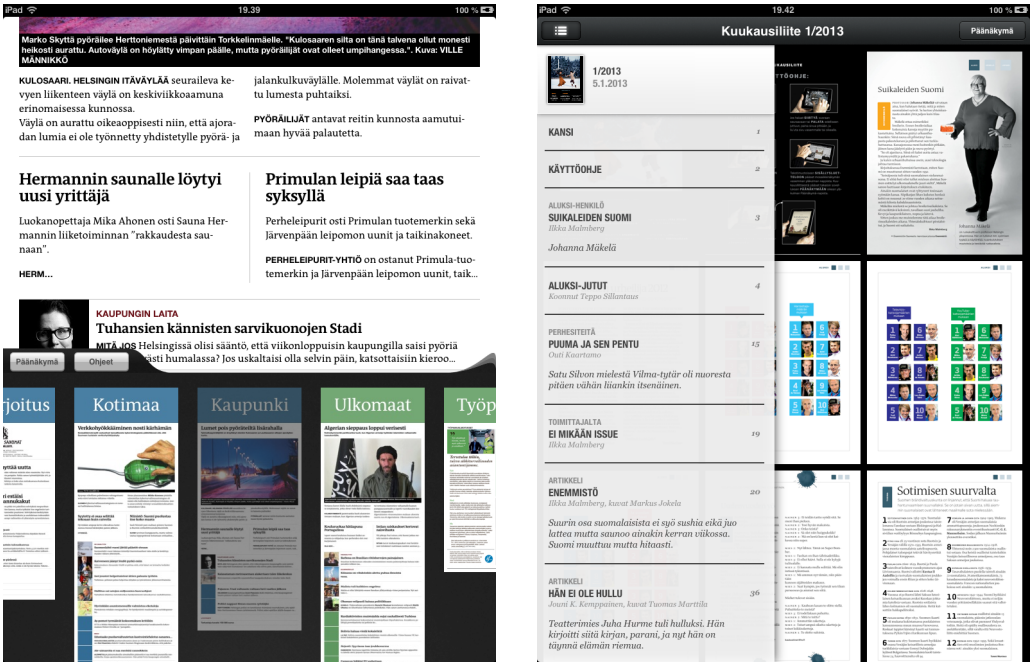
Vaikka Helsingin Sanomien iOS-sovelluksen päälehti ei ole näköislehti, se ei silti mukaudu responsiivisesti laitteen orientaatiomuutokseen, kuten kuvassa 4.5 nähdään. Saman sovelluksen sisällä julkaistava päälehti ja kuukausiliite eroavat merkittävästi toisistaan. Kuvassa 4.6 on havainnollistettu molempien lehtien navigaatioratkaisuja. Helsingin Sanomien kuukausiliite on näköislehti, mutta istuu rikkaan visuaalisen taittonsa ja hyvin käytettyjen eleiden vuoksi erinomaisesti tabletille. Käyttäjä voi pyyhkäistä sivuja normaalikoossa tavalliseen tapaan tai vaihtoehtoisesti nipistää itsensä sivujen esikatselukuvat listaavaan näkymään. Käyttäjälle on myös tarjolla artikkelilistaus lyhyillä kuvausteksteillä erillisessä valikossa. Helsingin Sanomien kuukausiliitteen kohdalla natiivin sovelluksen ja näköislehden tarjoama suorituskyky yhdistettynä hyvin toimivaan käyttöliittymään saa aikaan vakuuttavan kokonaisuuden. Valitettavasti sisältö ja käyttökokemus ei skaalaudu helposti muille laitteille. [52]

Kuva 4.5: Helsingin Sanomien iOS-sovelluksen selausnäkymässä sisältö ei täytä näyön koko leveyttä orientaation vaihtuessa. [52]

Sovellus lataa numerot laitteeseen käyttäjän ostaessa tai tilauksen ollessa voimassa koskettaessa uutta numeroa. Helsinkiläinen ohjelmistotalo SC5 on toteuttanut Sanomalle eri lehtien sovelluksien sisällä toimivan Tasku-palvelun, jossa käyttäjät voivat ostaa ja siirtyä Sanoman eri lehtiin[105]. Palvelun näkymät on toteutettu HTML5-tekniikalla, joten ne eivät ole riippuvaisia isäntäsovelluksen toteutuksesta ja niiden sisältöä voidaan päivittää itsenäisesti. Palvelu mahdollistaa lukijalle helpon siirtymisen julkaisijan tuoteportfolion sisällä, millä omalta osaltaan pyritään edistämään tuotteiden myyntiä[106].

[105] `http://tasku.fi/`

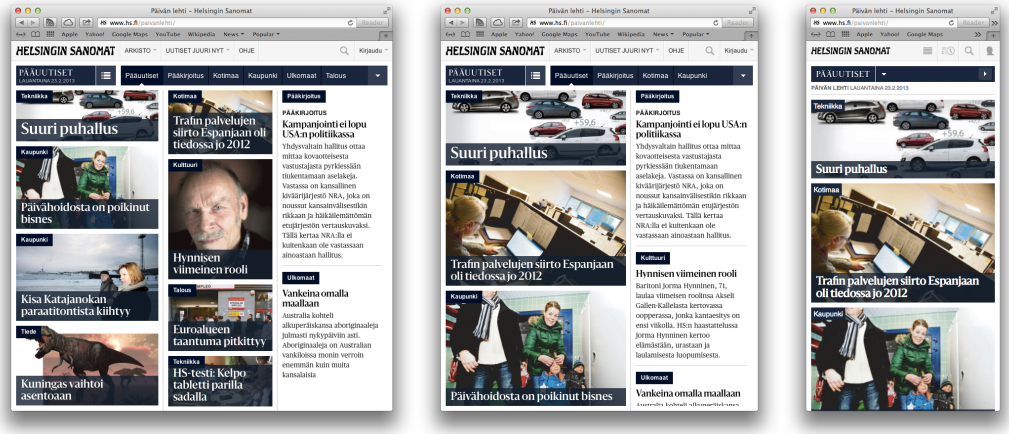
[106] `http://blog.sc5.fi/2012/01/sc5n-toteuttama-tasku-palvelu-julkaistiin/`



Kuva 4.6: Helsingin Sanomien päälehden ja kuukausiliitteen toteutukset eroavat merkittävästi toisistaan, saman sovelluksen sisällä. Kuukausiliitteessä (oikealla) käyttäjä pääsee sivut listaavaan esikatselunäkymään nipistyseleellä, jonka lisäksi tarjotaan erillisessä valikossa jutut listaava hakemisto. [52]

Helsingin Sanomilla on myös web-applikaatiototeutus [53], joka kulkee nimellä Päivän lehti. Web-applikaatio tukeutuu toteutuksessaan selainhistorian hyödyntämiseen, joten selattuja artikkeleita voidaan esimerkiksi lisätä kirjanmerkkeihin. Web-applikaatio tukee pyyhkäisyä, jota voidaan käyttää eri kategorioiden ja artikkeleiden välillä liikkumiseen. Sisältö haetaan reaaliaikaisesti käyttäjän pyynnöstä, jolloin esimerkiksi pyyhkäisyn aiheuttama artikkelinvaihto ei tapahdu välittömästi, vaan vasta uuden sisällön ollessa valmis näytettäväksi. Mikäli kuvat eivät ole ehtineet latautua, animoidaan ne esille läpinäkyvyyttä säätämällä latautumisen jälkeen. Sisällön animoinnissa hyödynnetään CSS-siirtymiä. Web-applikaatio mukautuu myös korkeussuunnassa laitteen näytölle, joten myös suurin osa artikkeleista sivutetaan. Sivujen välillä voidaan liikkua pyyhkäisemällä tai painikkeita käyttämällä. Web-applikaatiossa on hyödynnetty runsaasti responsiivista suunnittelua, joten se mukautuu merkittävästi eri resoluutioille. Mukautumista on havainnollistet-

tu kuvassa 4.7.



Kuva 4.7: Helsingin Sanomien Päivän lehti -web-applikaation ulkoasu mukautuu selaimen leveyteen. [53]

4.4 Suomen Kuvalehti

Suomen Kuvalehti on kunnostaunut digitaalisen julkaisunsa kanssa useammalla rintamalla. Lehdestä on olemassa iOS-alustalle oma hybridisovelluksena [54], mutta myös alustariippumaton web-applikaatio [55]. iOS-sovellus on esitetty kuvassa 4.8 ja web-applikaatio kuvassa 4.9. Teknisesti mielenkiintoisen Suomen Kuvalehden sovelluksista tekee tosiasia, että käyttäjän on vaikea erottaa pelkän selaamansa artikkelin perusteella onko kyseessä natiivi sovellus vai web-applikaatio. Molemmissa sovelluksissa julkaistu sisältö on täsmälleen samassa formaatissa ja lehden numeroiden taitto sisältää sekä staattista, että responsiivista sisältöä. Yhden sivun kuvat tai mainokset on otettu mukaan staattisina kuvina, mutta vähänkin pidemmät artikkelit on taitettu responsiivisesti tarjoten eri laitteille mukautuvan lukukokemuksen.

Kuva 4.8: Suomen Kuvalehden iOS-sovelluksella selattu artikkeli ja sen päälle avautunut navigaatiovalikko. [54]

Suomen Kuvalehden web-applikaation toteutuksessa on käytetty jälleen kerran CSS3:n 3D-transformaatiota [55]. Sivujen välillä liikutaan pyyhkäisyllä, joka muuttaa sivut sisältävän isäntäelementin sijaintia vaakasuunnassa [55]. Sivujen sisällön vierityksessä ei ole käytetty CSS3-tekniikkaa vaan tavallista sisällön vieritystä [55]. Esimerkiksi vierekkäisten palstojen sisältöä selataan erikseen [55]. Tavalliseen vieritykseen tukeutuminen aiheuttaa ongelmia erityisesti Android Browser -selaimessa Android 3.0 -versiota vanhemmilla alustoilla, jotka eivät tue CSS:n overflow-asetuksen auto tai scroll -arvoja[107]. Alusta- ja selainkannan uusiutuessa Suomen Kuvalehden ratkaisusta tulee kuitenkin käyttökelpoisempi, jolloin myös sovelluksen kehittäminen helpottuu, sillä kehittäjien ei tarvitse huolehtia vierityksen teknisestä toiminnasta.

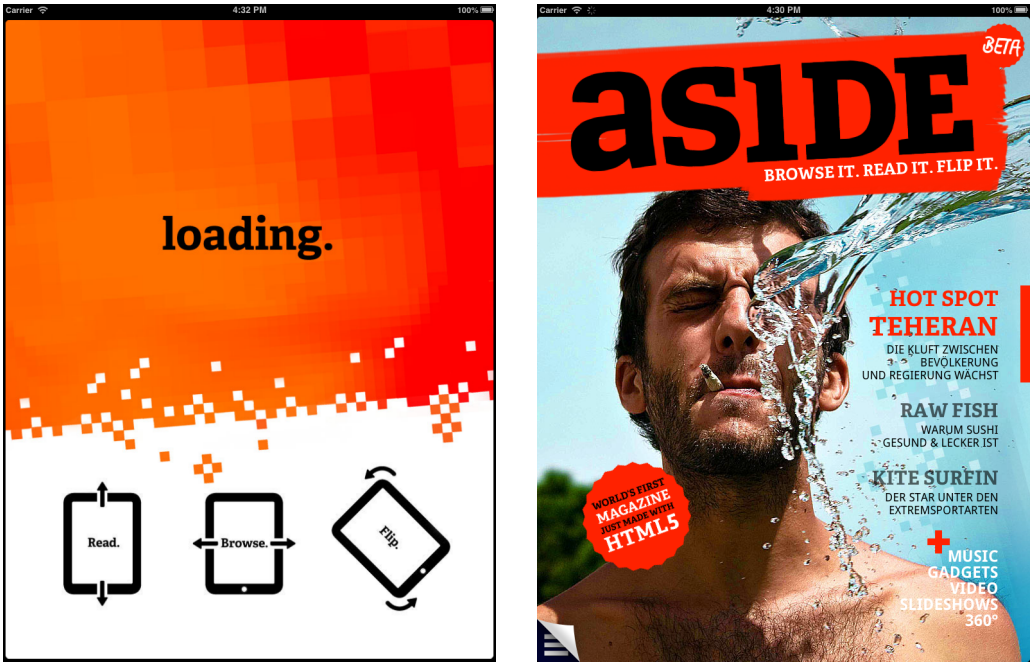
[107] `http://barrow.io/overflow-scrolling/`

Kuva 4.9: Suomen Kuvalehden digitaalisen lehden sisältö sisältää suurimmaksi osaksi responsiivisesti taitettua ulkoasua. Sisällön ollessa täsmälleen sama on näiden kuvien perusteella mahdotonta kertoa onko kyseessä Suomen Kuvalehden iOS-sovellus vai web-applikaatio. [55]

Suomen Kuvalehti käyttää sovelluksissaan luvussa 3.5.3 esiteltyä PhotoSwipe-kirjastoa, jonka avulla artikkeleiden kuvat voidaan avata erilliseen kuvagalleriaan. Kuvagalleria mahdollistaa suurempien kuvien tarjoamisen pyynnöstä ja antaa käyttäjälle mahdollisuuden suurentaa kuvaa. Kuvagalleriaan ei kuitenkaan pääse mistä tahansa kuvasta, vaan valikoiduista infograafeista ja artikkelin lopussa sijaitsevista pienistä esikatselukuvista. [54, 55]

4.5 Aside Magazine

Aside Magazine [56] on ainoastaan verkossa julkaistu kokonaan HTML5-teknologioilla tehty lehti, joka on pakattu visuaalisesti näyttäväksi web-applikaatioksi. Lehdestä on julkaistu diplomityön kirjoittamisen aikana ainoastaan yksi numero demonstroimaan käytettyä teknologiaa. Lehti sisältää artikkeleiden lisäksi muun muassa ääntä, videota ja animoituja infograafeja, joista viimeksi mainitut tarjoavat ilmeen elävöitymisen lisäksi palstatilan säästöä, sillä yhdellä paikalla voidaan animaatioiden avulla esittää vaihtuvaa sisältöä. Aside Magazine on esitetty kuvissa 4.10 ja 4.11.

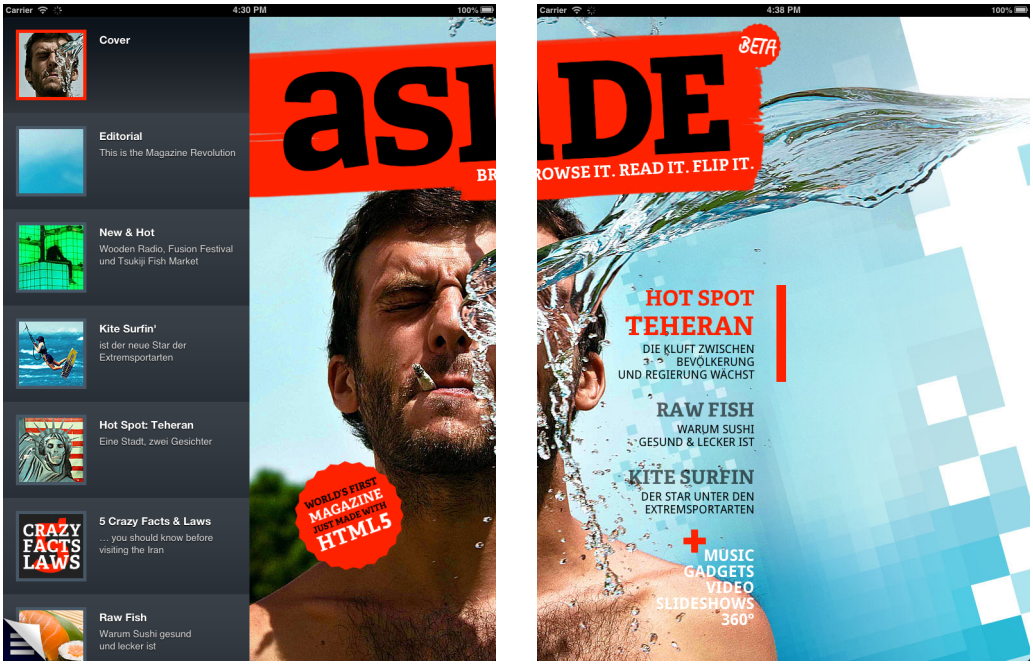


Kuva 4.10: Aside Magazine -julkaisun käynnistyskuva ja etusivu. Käyttäjä ohjaa lehteä pyyhkäisyllä ja vierityksellä. Myös orientaatiota voidaan vaihtaa. [56]

Aside Magazine mainostaa itseään maailman ensimmäisenä kokonaan HTML5:llä toteutettuna lehtenä - diplomityön kirjoittamisen aikana lehtiä on kuitenkin julkaistu jo enemmän, esimerkkinä kotimainen HBL+[108]. Teknisesti Asiden selausnäkymässä on kuitenkin turvauduttu perinteisempään toteutustapaan verrattuna esimerkiksi Financial Timesin web-applikaatioon: pyyhkäisy muuttaa sivujen DOM-elementeille asetettua marginaalia CSS3:n 3D-transformaation sijasta, mikä tekee sivujen selaamisesta suorituskyvyltään kankeampaa 3D-kiihdytyksen puuttuessa - vasta pyyhkäisyn kosketuksen loppuessa suoritetaan loppusiirtymä CSS3-animaatiolla. Suorituskykyero havainnollistuu voimakkaammin pikselitiheydeltään suuremmassa Retina-laitteessa. Julkaisun kuvia ei myöskään ole optimoitu suuremmille pikselitiheyksille, mikä puolustaisi teknologiademona toimivan lehden suunnittelun keskittyneen ensisijaisesti aikaisemman sukupolven laitteille. Lehden elävä taitto tekee siitä kuitenkin laitteesta riippumatta yhden näyttävimmistä HTML5-taitetuista lehdistä. Aside Magazinen tekijät ovat julkaisseet

[108] `http://plus.hbl.fi`

myös oman CSS-kirjaston Magazine Grid[109], joka helpottaa HTML5-lehden ulkoasun teknistä toteuttamista. [56]



Kuva 4.11: Aside Magazinen navigaatiovalikko aukeaa lehden vasemman alareunan taitetusta kulmasta ja listaa kaikki lehden sivut lyhyellä nostolla. Aside Magazinen etusivun taustakuva jatkuu seuraavan sivulle, mikä piristää taittoa omalta osaltaan. [56]

[109] `http://grid.asidemag.com/`

Luku 5

Toteutukset

Tässä luvussa käydään lävitse diplomityön aikana toteutetut sovellukset ja demolehtenä toimineen Aalto University Magazine -lehden tablettiversion toteutus. Lisäksi luvun lopussa käsitellään diplomityön tuloksena syntynyttä Stage Framework -ohjelmistokehystä, jonka osana toteutettu web-applikaatio toimii. Suurin osa diplomityöhön liittyneestä käytännön työstä liittyi demolehden ja sovellusten implementointiin. Työn kokeelliseen osuuteen liittyvät alusta- ja selainkartoitus sekä suorituskykymittaukset on dokumentoitu tarkemmin luvussa 6.

5.1 Taustaa

Diplomityön osana toteutettiin demolehti Aalto University Magazine -lehden tablettiversioksi. Demolehti palveli pohjana HTML5-taitetun sisällön jakelukanavien tutkimiselle, joista diplomityössä sovellettiin kahta. Lehdelle toteutettiin natiivi sovellus iOS-alustalle wrapper-sovelluksena, joka julkaistiin myöhemmin Aalto University Magazinen tabletti- ja mobiilikonseptina vapaasti ladattavaksi Applen App Storessa[110]. Lisäksi lehdelle toteutettiin web-applikaatio, jonka toteutuksessa etsittiin parhaita käytäntöjä alustariippumattomalle digitaaliselle julkaisemiselle. Web-applikaatiosta kehittyi diplomityön aikana oma ohjelmistokehys, Stage Framework, joka tarjoaa yhden sivun sovelluksen lehtihyllyllä digitaalisten julkaisujen tarjoamiseksi web-applikaationa, pohjautuen HTML5-muotoiseen sisältöön. Seuraavassa on esitelty toteutetun demolehden ja sovellusten toteutuslähtökohdat, Aalto University Magazine -lehti, sekä demolehden suunnittelussa käytetty konsep-

[110] http://itunes.apple.com/fi/app/aalto-university-magazine/id596682061/

tikäsikirja.

5.1.1 Toteutusten lähtökohdat

Sovelluksien toteutuksen tutkimuksellinen lähtökohta oli vertailla HTML5-taitetun sisällön käyttöä natiivin sovelluksen ja web-applikaation osana. Premissinä oli, että HTML5-taitettua sisältöä voidaan jaella alustariippumattomasti natiiveja sovelluksia lähestyvällä suorituskyvyllä käyttäen hyödyksi uusia selainteknologioita, joita HTML5 ja CSS3 molemmat tarjoavat. Teknisten ratkaisujen lisäksi keskeisessä asemassa oli sisältö ja sen tuottaminen automatisoidusti. Sisällön suunnittelussa painotettiin responsiivista suunnittelua ja sisällöntuotannon älykästä automatisointia, kuten kuvien rajausta sisältöön perustuen ja lehden ulkoasun mukautumista sen sisältöön. Visuaalisen median tutkimusryhmässä Next Media -projektin osana on tutkittu aikaisemmin aiheeseen liittyen esteettisiä mittoja automaattisesti taitetuille tablet-aikakauslehdille [57] sekä tablettien designratkaisujen käytettävyysarviointia [58].

Toteutettu demolehti palveli myös Aalto University Magazine -lehteä konkretisoimalla suunnitellun tablettilehtikonseptin. Diplomityössä haluttiin kartoittaa mobiilialustojen valmiutta eri selainteknologioille ja niiden soveltamiselle web-applikaation osana. Toisaalta diplomityössä oltiin myös kiinnostuneita selainteknologioiden suorituskyvystä ja niiden suhteesta natiivilla wrapperilla toteutetun sovelluksen suorituskykyyn. Web-applikaatiototeutukselta odotettiin myös tukea natiiveista sovelluksista tutulle pyyhkäisyeleelle, jolla sivujen välillä voidaan liikkua.

5.1.2 Aalto University Magazine

Aalto University Magazine on Aalto-yliopiston viestinnän julkaisema neljä kertaa vuodessa ilmestyvä sidosryhmälehti, joka kertoo tieteestä ja taiteesta, tekniikasta ja taloudesta - sekä niiden kohtaamisesta ja ihmisistä tarinoiden takana[111]. Lehti julkaistaan painettuna, mutta siitä on saatavilla näköislehti lehden verkkosivustolla. Aalto-yliopiston viestintä tuli mukaan Next Media -projektiin tarkoituksena kartoittaa mahdollisuutta julkaista lehdestä versio tabletti- ja mobiililaitteille. Lehdestä toteutettiin luvussa 5.2 kuvattu demolehti perustuen konseptikäsikirjaan, josta kerrotaan tarkemmin luvussa 5.1.3.

Aalto-yliopiston viestintä toimitti demolehteä varten suomen- ja englan-

[111] http://aalto.fi/fi/current/magazine/

ninkielisiä versioita artikkeleista, jotka oli julkaistu aikaisemmin lehden kolmannessa painetussa numerossa. Painetussa lehdessä on sekä suomen- että englanninkielisiä artikkeleita, joten demolehteä varten haluttiin toimittaa versiot molemmilla kielillä mahdollisimman monesta artikkelista. Lisäksi demolehteä varten toimitettiin täydentävää video- ja audio-sisältöä artikkeleihin liittyen. Aalto-yliopiston viestinnän yhteyshenkilöinä Next Media -projektissa toimivat toimituspäällikkö Paula Haikarainen ja päätoimittaja Eveliina Olsson.

5.1.3 Konseptikäsikirja

Demolehden ulkoasun suunnitteli Juho Hiilivirta, joka toteutti kurssityönä Aalto University Magazinen tabletti- ja mobiiliversion konseptikäsikirjan [59]. Hiilivirta on ollut mukana toteuttamassa myös Aalto University Magazinen painetun lehden visuaalista ilmettä. Lehti toteutettiin teknisesti diplomityön aikana konseptikäsikirjan pohjalta yhteistyössä diplomityön ohjaajan Mikko Kuhnan kanssa.

Konseptikäsikirjan Aalto University Magazine eroaa selkeästi painetusta lehdestä, jolla omalta osaltaan pyritään erottautumaan näköislehtityyppisestä julkaisusta. Konseptikäsikirjassa kuvataan lehden aloituskuvat, sisältösivu (etusivu), artikkelisivu ja niiden käyttäytyminen erikokoisilla laitteilla ja laiteorientaatioilla. Lisäksi määritellään muun muassa lehdessä käytettävät kirjasimet ja tehostevärit. Kuvassa 5.1 on esitetty ote konseptikäsikirjasta, jossa on määritelty artikkelisivun rakenne, julkaisussa käytettävät tehostevärit ja kirjasintyypit, sekä sisältösivun käyttäytyminen. [59]

Kuva 5.1: Konseptikäsikirja kuvaa visuaalisen ilmeen lisäksi sisällön asettelun ja käyttäytymisen eri kokoisilla laitteilla. [59]

Konseptikäsikirjassa lehdelle ei ole suunniteltu varsinaisia käyttöohjeita, vaan lehden suunnittelussa on pyritty huomioimaan lukemista auttavat visuaaliset vihjeet, kuten jutun otsikon näkyminen ja optimaalinen leikkaantuminen artikkelin ison pääkuvan alla. Tällä tavalla lukija huomaa sisällön jatkuvan luonnollisesti ja osaa alustakohtaisten käyttökonventioiden avulla vierittää sivua. Kuvassa 5.2 on esitetty konseptikäsikirjan havainnollistamana sisällön jatkuvuus ja kuvien rajaus eri laiteorientaatioilla. [59]

Kuva 5.2: Konseptikäsikirja määrittelee myös muun muassa visuaalisen vihjeen sisällön jatkuvuudesta. [59]

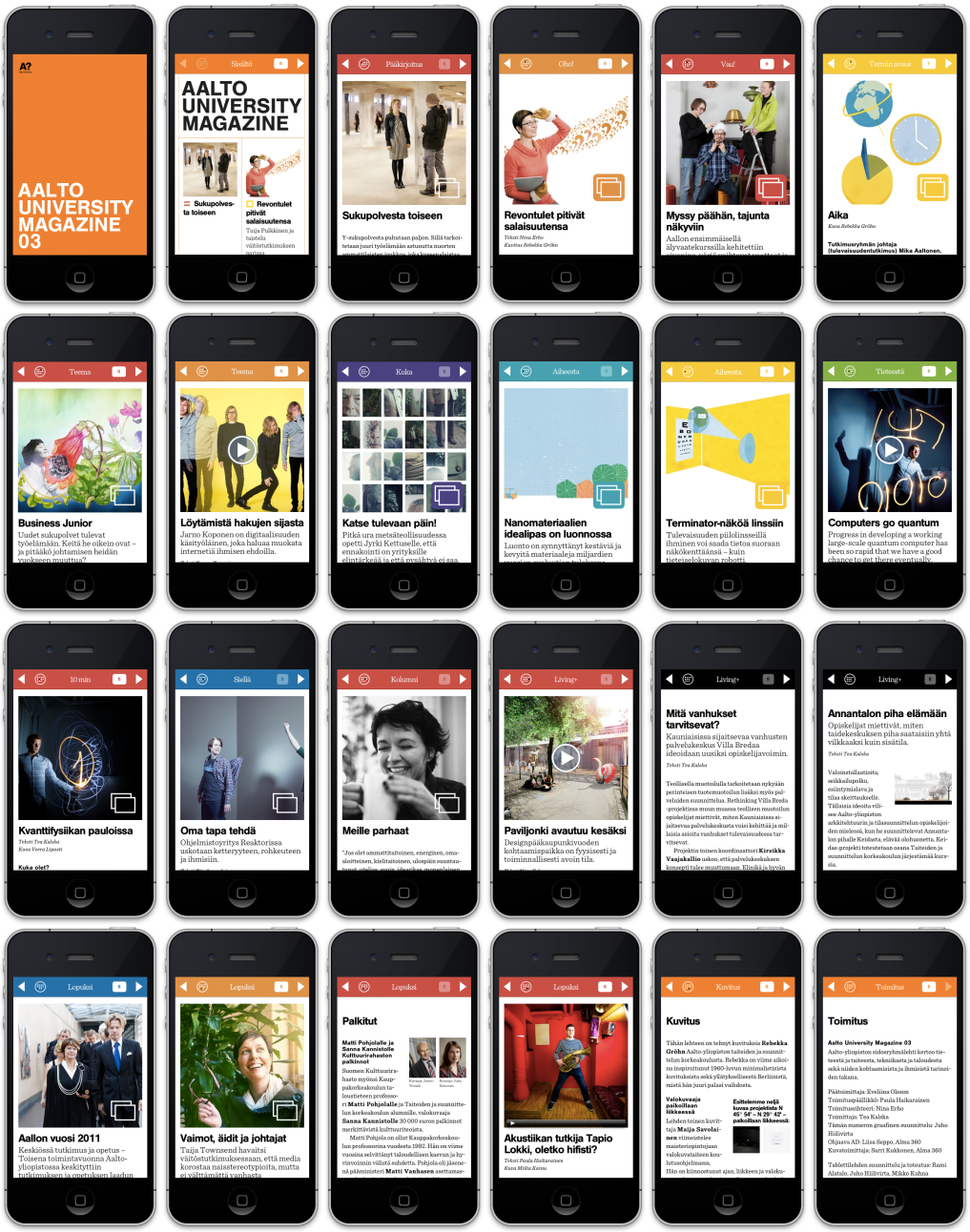
5.2 Demolehti

Tässä alaluvussa on esitelty Aalto University Magazine -demolehden ulkoasu ja käyttöliittymä, sekä tehtyjä teknologisia ratkaisuja. Alaluvun lopuksi esitellään tablettiversion julkaisuprosessi ja siihen liittyvät resurssit. Demolehti on toteutettu pohjautuen aikaisemmin kuvattuun Aalto University Magazine -lehden tablettiversion konseptikäsikirjaan. Lehti toteutettiin diplomityön aluksi ennen toteutettua iOS-sovellusta ja web-applikaatiota.

5.2.1 Ulkoasu ja käyttöliittymä

Demolehden ulkoasun toteuttamisessa tukeuduttiin responsiiviseen suunnitteluun ja CSS-mediakyselyihin. Kohdealustoiksi rajattiin toteutuksen aikana tabletti- ja mobiilialustat, sillä lehden suunnittelu samanaikaisesti työpöytäkäyttöön olisi monimutkaistanut tukea erityisesti näytön leveydeltään ja pikselitiheydeltään kasvaville tableteille, jotka ottavat kiinni pienempiä työpöytäkoneita ja kannettavia tietokoneita. Myös pikselitiheydeltään suuremmat matkapuhelimet haastavat ja jopa ohittavat tavallisella pikselitiheydellä varustettujen tablettien resoluutiot. Haastavimmaksi responsiivisen suunnittelun kannalta muodostuu kuitenkin työpöytä- ja tablettilaitteen erottaminen pelkkien CSS-mediakyselyiden avulla. Mediakyselyiden toimintaa demolehdessä on esitelty tarkemmin seuraavassa alaluvussa 5.2.2.

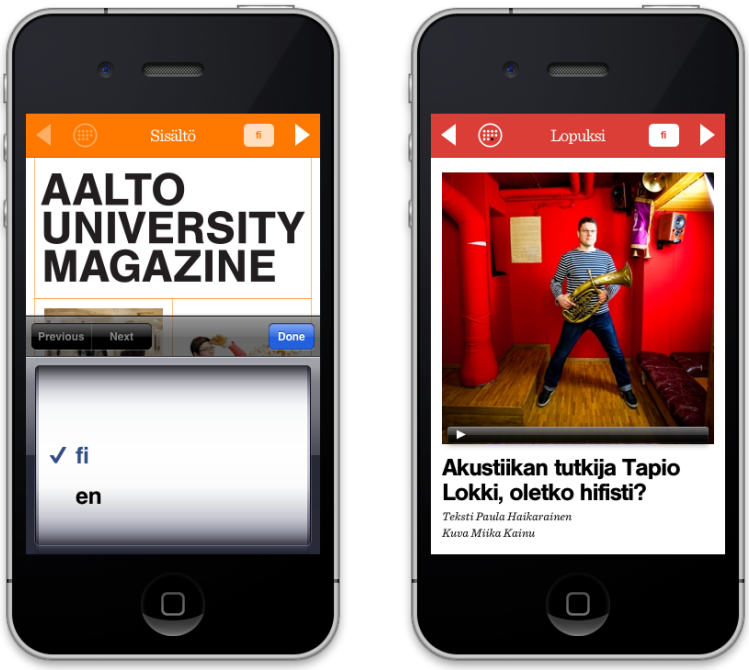
Lisäyksenä konseptikäsikirjaan demolehteen toteutettiin artikkelin pääkuvan mukaan vaihtuva tehosteväri, jota on havainnollistettu kaikki lehden näkymät ja artikkelit listaavassa kuvassa 5.3. Tehosteväri valitaan konseptikäsikirjan määrittelemien yhdentoista tehostevärin joukosta, jotka oli alkuperäisessä suunnitelmassa tarkoitettu numerokohtaisiksi. Tehosteväri vaikuttaa navigaatiopalkin, kuvagalleria-ikonin ja artikkelin nostotekstien väriin. Kuvagalleria-ikonissa tehosteväriä ei käytetä artikkeleissa, joissa valkoinen ikoni erottuu taustasta ilmankin. Ikoni ei myöskään ole läsnä artikkeleissa, jotka sisältävät pääkuvan päältä aktivoitavaa videota tai ääntä. Toteutettuun artikkelikohtaiseen kuvagalleriaan pääsee kuitenkin kaikista artikkelin kuvista.



Kuva 5.3: Aalto University Magazine -demolehden aloituskuva ja sivut esitettynä iOS-alustalla.

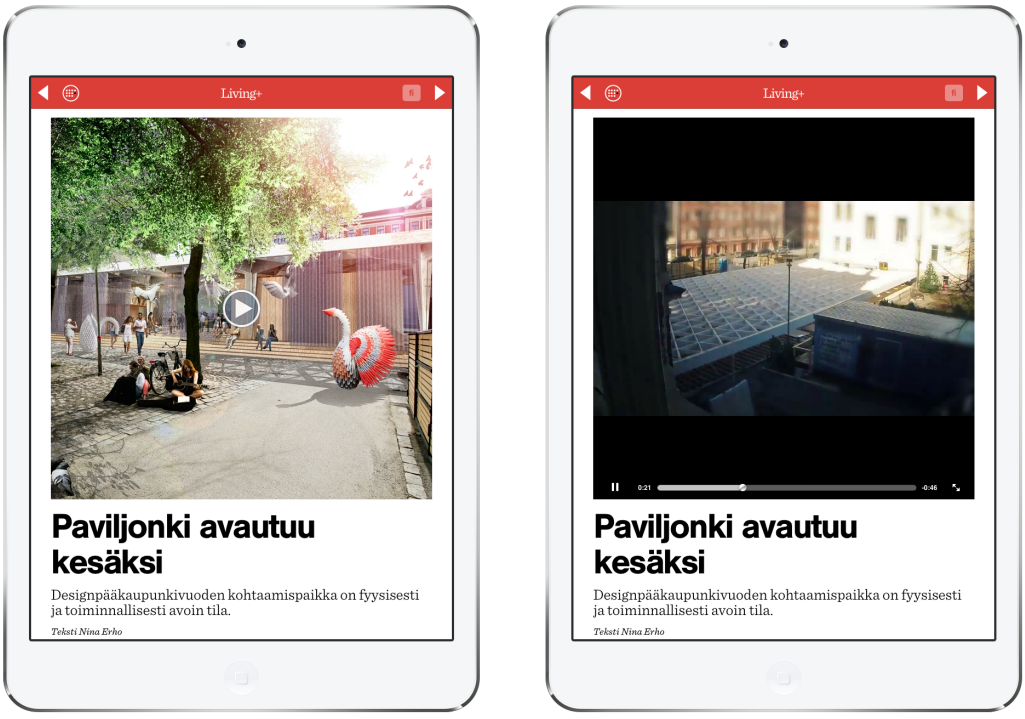
Lehdelle toteutettiin kuvassa 5.3 ensimmäisenä näkyvä aloituskuva, joka näytetään sitä tukevissa, pääasiassa iOS-alustan laitteissa. Lehden sisältö-

sivulle (kuvassa 5.3 toisena) lisättiinkin myöhemmässä vaiheessa Aalto-yliopiston viestinnältä saadun palautteen ja luvussa 5.3.3 kuvatun käytettävyysarvioinnin perusteella Aalto University Magazine -otsaketeksti. Lehti koostuu sisältösivun lisäksi artikkelisivuista, jotka on sijoitettu painetun lehden järjestystä mukaillen peräjälkeen sisältösivusta eteenpäin. Jokaisella sivulla on sivun yläosaan sijoitettu navigaatiopalkki, joka pysyy kiinni näytön yläosassa vaikka sisältöä vieritettäisiin - se on siis aina näkyvissä. Navigaatiopalkki sisältää nuolet vasemmalle ja oikealle, joista käyttäjä voi siirtyä järjestyksessä edeltävälle ja seuraavalle sivulle, erityisesti jos käytössä ei ole pyyhkäisyelettä, kuten ilman erityistä ohjelmistokehystä toimivassa web-applikaatiossa. Navigaatiopalkki sisältää myös linkin sisältösivulle, joka havainnollistaa myös nykyisen sivun sijaintia lehdessä artikkelin järjestyksen perusteella valitulla ikonilla. Lisäksi käyttäjä pystyy valitsemaan kielen sisältösivulla ja niissä artikkeleissa, joista on saatavilla useampia kieliversioita. Kielipainikkeen painaminen avaa alusta- ja selainkohtaisen valitsimen (kuvassa 5.4 vasemmalla). Artikkeleissa, jotka eivät tarjoa useampia kieliversioita, kielipainikkeen sävy on himmennetty.



Kuva 5.4: Vasemmalla natiivi kielivalintavalikko, oikealla mediasoitin äänelle iOS-alustalla.

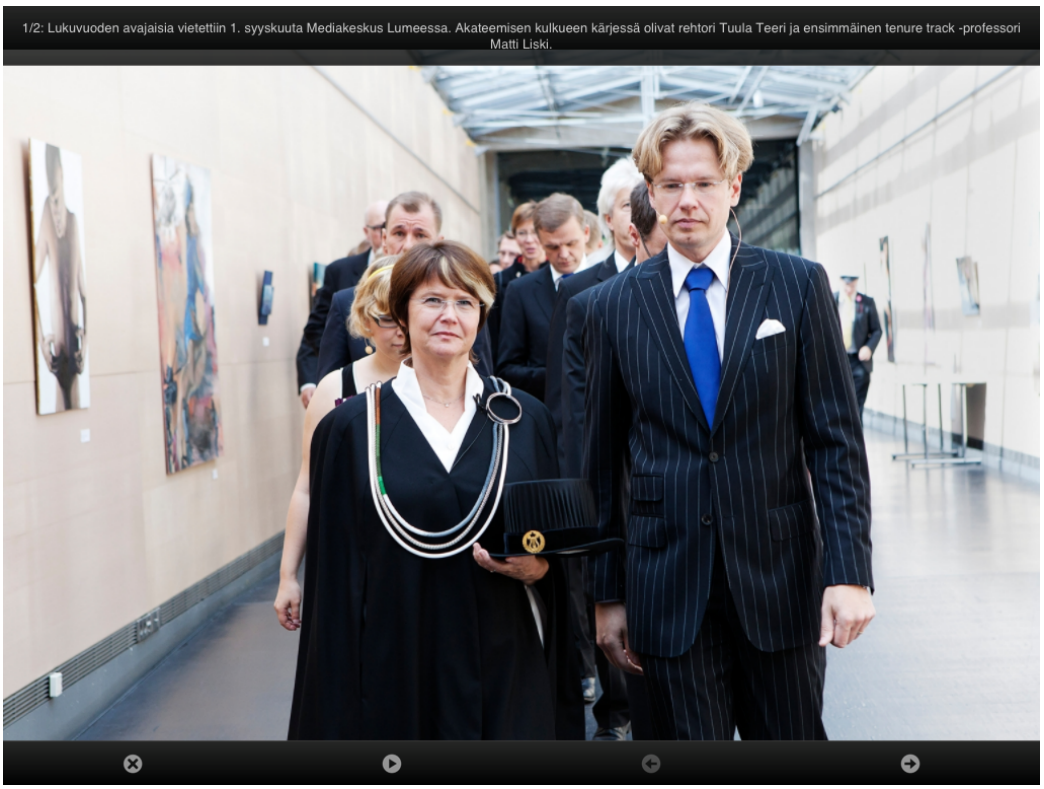
Demolehdessä julkaistiin lisämateriaalia painettuun lehteen videon ja äänen muodossa. Lisämateriaali sijoitettiin sisältöön käyttäen HTML5:n *video*- ja *audio*-elementtejä, jotka niin ikään käyttävät alusta- ja selainkohtaisia käyttöliittymiä. Äänitiedoston tarjoaminen iOS-alustalla on esitetty kuvassa 5.4 oikealla: pääkuvan alaosaan ilmestyy iOS-alustalle ominainen äänisoitin. Soittimen visuaalinen ilme vaihtelee käytettävän selaimen ja alustan mukaan. Myös täysin oman käyttöliittymän toteuttaminen HTML5-mediaelementeille on mahdollista. Kuvassa 5.5 on esitetty video-elementin toiminta iOS-alustalla iPad-laitteella: artikkelin pääkuva toimii videon aktivoivana alueena (kuvassa vasemmalla) ja videon alareunaan ilmestyvät iOS:lle ominaiset videonhallintasäätimet (oikealla).



Kuva 5.5: Videoelementin toiminta iOS-alustalla.

Demolehdessä käytettiin PhotoSwipe-kirjastoa toteuttamaan artikkelikohtainen kuvagalleria. Vaikka artikkeleissa käytetyt kuvat rajattiin eri resoluutioille sopiviksi, kuvagallerian kautta käyttäjä pääsee aina käsiksi täysilaatuisiin kuviin, joita voidaan esimerkiksi suurentaa PhotoSwipen käyttöliittymässä nipistys-eleellä. Muualla demolehden käyttöliittymässä sisällön skaalaus on

poistettu käytöstä, sillä sisältö mukautetaan laitteelle sopivaksi responsiivisella suunnittelulla. Kuvagalleria on toteutettu täysilaatuisten kuvien tarjoamisen vuoksi myös artikkeleille, joissa on ainoastaan yksi kuva. Gallerian käyttöliittymässä näytetään kuvaan liittyvä kuvateksti, jota ei välttämättä näytetä artikkelissa pienemmillä resoluutioilla tilan säästämiseksi. Kuvatekstin yhteydessä on myös kuvan järjestysnumero artikkelissa ja kuvien kokonaismäärä, jotta käyttäjä pystyy seuraamaan etenemistään kuvagalleriassa. PhotoSwipella toteutettu kuvagalleria on esitetty kuvassa 5.6.



Kuva 5.6: PhotoSwipe-kirjastolla toteutetun kuvagallerian käyttöliittymä: yläreunassa kuvan järjestysnumero, kokonaiskuvamäärä ja kuvateksti; alareunassa kuvagallerian sulkupainike, kuvaesitys-toiminnallisuus ja siirtymispainikkeet. Käyttäjä voi myös pyyhkäistä kuvasta toiseen.

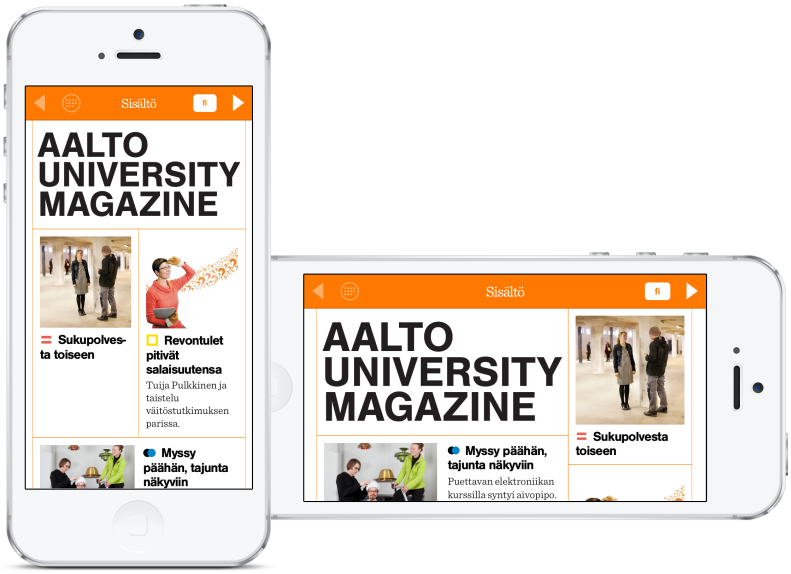
Lehden artikkelin ulkoasu perustuu navigaatiopalkin ja mahdollisen pääkuvan lisäksi pääpalstaan ja sen vierelle sijoitettaviin kainaloteksteihin. Pääpalstan ja kainalotekstien sekaan voidaan sijoittaa lisäkuvia, joita voi kainalotekstissä olla kaksi vierekkäin. Lisäksi pääpalstan leipätekstiä voidaan

jäsentää yhden tason alaotsikoilla sekä erityisillä nostoteksteillä. Artikkeleissa on myös pääsääntöisesti pääkuvan alapuolelle sijoitettu ingressi ja tekijöiden nimet. Kaikki artikkelin elementit noudattavat konseptikäsikirjassa määriteltyä visuaalista ilmettä.

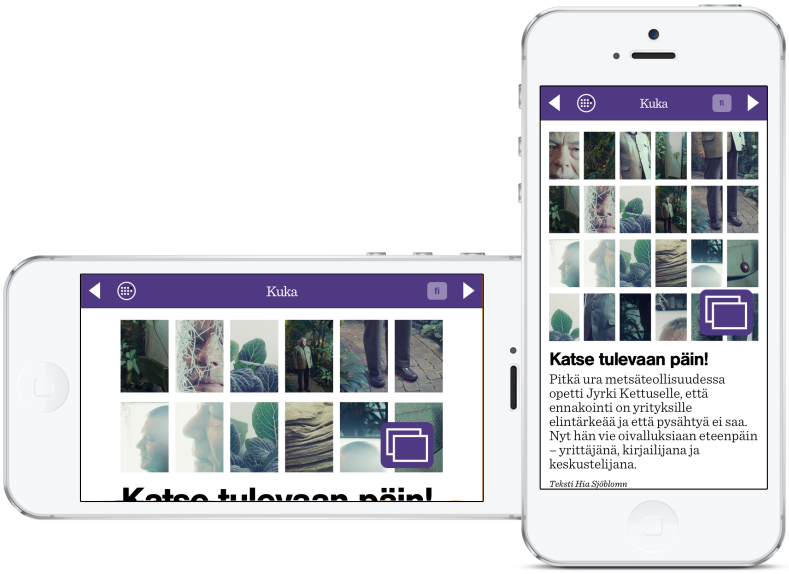
5.2.2 Tekniset ratkaisut

Demolehden toteutuksessa on tukeuduttu responsiiviseen suunnitteluun ja CSS-mediakyselyiden käyttöön. Mediakyselyitä käytetään hieman eri tavoin sisältösivulla ja artikkelisivulla. Artikkelisivulla sisällön mukautuminen selainnäkymän leveyteen etenee askeleittain. Sisältösivu mukautuu lineaarisesti, mutta sisältää kolme eri asettelua, jotka perustuvat joko yhteen tai kahteen palstaan. Pääsääntöisesti pienemmille resoluutioille tarjotaan yhden palstan asettelu ja suuremmille kaksi. Sisältösivun nostotekstityylillä esitetyt artikkelit sijoitetaan kaksipalstaisessa ulkoasussa joko apupalstaan tai pääpalstaan riippuen selainnäkymän leveydestä.

Palstajaon valinta riippuu kuitenkin muustakin kuin leveydestä: konseptikäsikirjassa iPhone-laitteen vaakatason orientaatiolle on määritelty kaksi palstaa, kun taas pystyorientaatiossa palstoja on yksi [59]. Pikselitiheydeltään suuremman Retina-näytöllisen iPhone-laitteen pystyorientaation pikselileveys on kuitenkin suurempi (640px) verrattuna ei-Retinallisen iPhone-laitteen vaakaorientaatioon (480px). Asettelun valinnassa joudutaan siis regressiivisesti pudottamaan palstajako kahdesta yhteen, jos tarkastellaan pelkästään selainnäkymän leveyttä. CSS-mediakyselyt mahdollistavat kuitenkin myös orientaation tiedustelemisen ja iOS-alustalla tieto saadaan myös pikselitiheydestä, jota ei ole mukana CSS-mediakyselyspesifikaatiossa. Sisällön mukautumista on havainnollistettu kuvissa 5.7 ja 5.8 (iPhone 5) sekä 5.9 (iPad-laitteet).

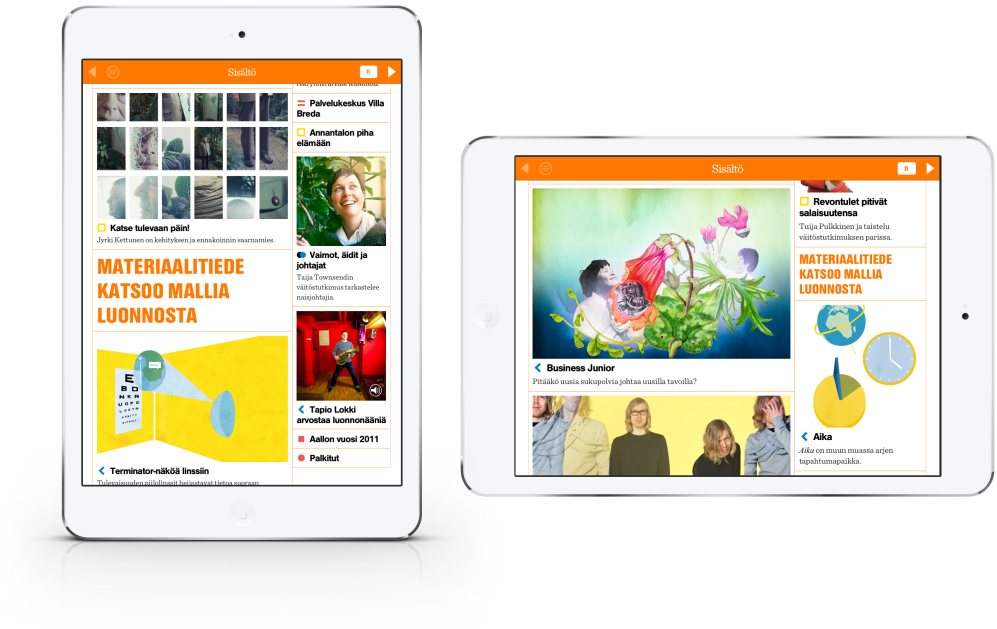


Kuva 5.7: Demolehden sisällön mukautuminen orientaatioon iPhone 5 -laitteella, sisältösivu.



Kuva 5.8: Demolehden sisällön mukautuminen orientaatioon iPhone 5 -laitteella, artikkelisivu.

Kuvassa 5.8 havainnollistettuun sisällön mukautumiseen orientaatiomuutokseen liittyy keskeisesti artikkeleiden pääkuvien rajaaminen eri tavoin: artikkelin tekstisisältö näkyy kaikissa tilanteissa, jotta käyttäjä huomaa sisällön jatkuvan alaspäin. Kaikista lehdessä käytetyistä kuvista ja niiden eri rajauksista on toteutettu versiot myös Retina-näytöille.

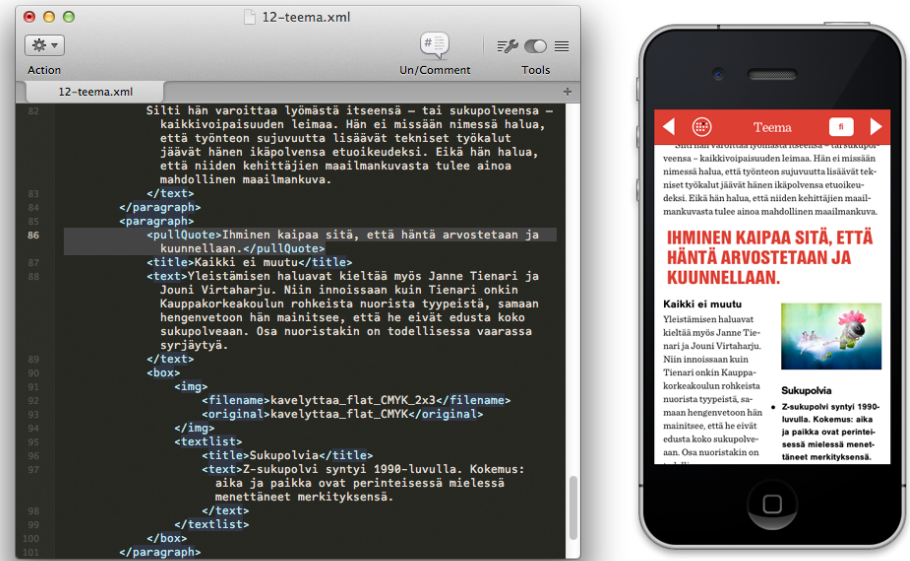


Kuva 5.9: Demolehden sisältösivu iPad-laitteen eri orientaatioilla.

Demolehdessä käytettiin web-kirjasimia kaikissa saatavilla olevissa formaateissa, jotta tuki olisi mahdollisimman laajaa. Käytettävät kirjasimet, Nimbus Sans Bold ja Sentinel, oli määritelty tablettilehden konseptikäsikirjassa. [59] Lehdessä tarjottujen lisäsisältöjen osalta formaatteja tuettiin rajoitetummin: esimerkiksi videot tarjottiin H.264/MPEG-4 AVC ja VP8 -pakattuina MP4 ja WebM -säiliöformaateissa. Lehden käyttöliittymäelementeissä, kuten navigaatiopalkissa käytettiin SVG-ikoneita niissä laitteissa, joissa SVG:tä tuettiin. SVG-tuki tarkistettiin sivun lataamisen yhteydessä JavaScriptillä käyttäen Modernizr-kirjastoa. Modernizr lisää automaattisesti *html*-elementille *svg*-luokan, jonka perusteella ikoneita taustakuvina käyttäville elementeille voidaan CSS-tyylimäärittelyissä ottaa käyttöön SVG-resurssit. Muussa tapauksessa tukeuduttiin PNG-resursseihin.

5.2.3 Julkaisuprosessi

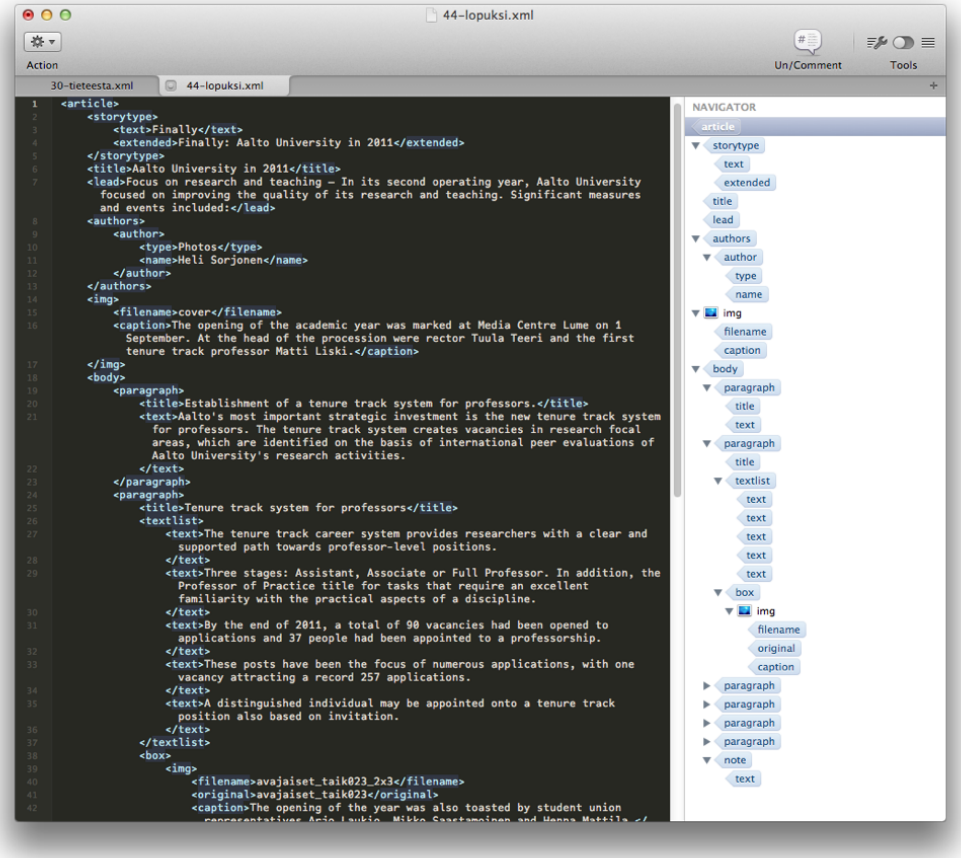
Aalto University Magazine -demolehden tapauksessa julkaisuprosessi lähtee liikkeelle painetun lehden digitaalisesta taittotiedostosta eli InDesign[112]-dokumentista. Tavoitteena kuitenkin oli, että tablettilehden julkaisu voitaisiin integroida esimerkiksi sähköiseen julkaisujärjestelmään, joten geneeriseksi lähdeformaatiksi valittiin XML[113]. InDesign-sovelluksella taitetut artikkelit siirretään siis manuaalisesti XML-formaattiin tai resurssit tuotetaan automaattisesti valitusta julkaisujärjestelmästä. Visuaalisen median tutkimusryhmässä on myös kartoitettu Next Media -projektin osana aikakauslehden artikkelin formaalia rakennetta[114] ja määritelty julkaisuprosessiin soveltuva sisällönkuvailumalli, jota voitiin käyttää artikkeleiden XML-resursseja tuotettaessa. Kuvassa 5.11 on havainnollistettu artikkelissa sijaitsevan nostotekstin muuttumista XML:stä HTML5:ksi.



Kuva 5.10: Nostotekstin siirtyminen XML:stä HTML5-muotoiseen artikkeliin.

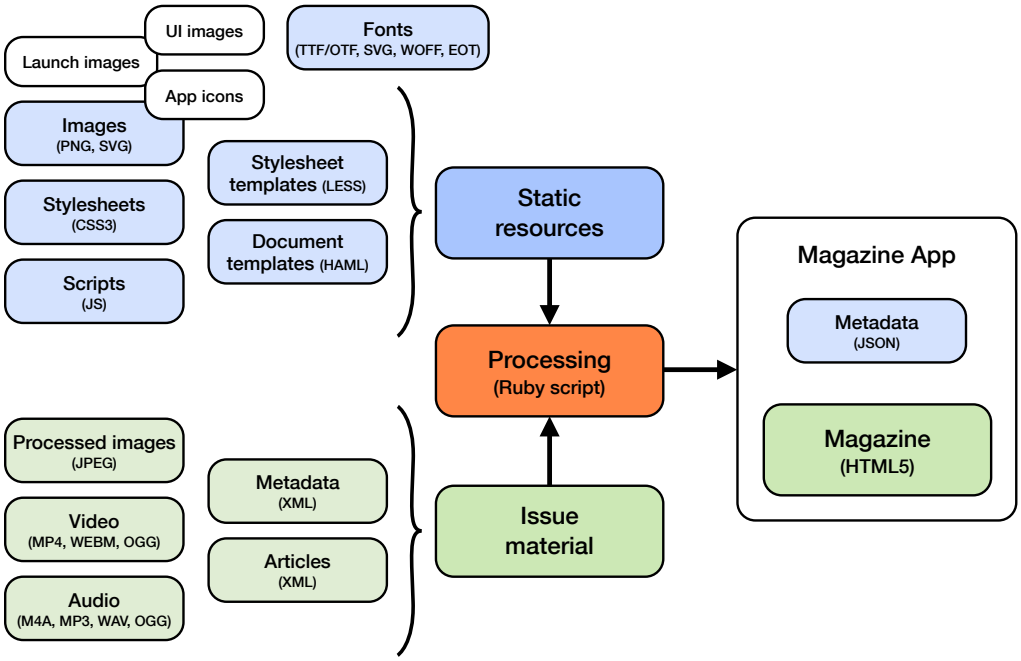
[112] http://adobe.com/products/indesign.html
[113] http://w3.org/TR/REC-xml/
[114] http://mikkokuhna.com/researcher/model/

Jokainen artikkeli koostui siis omasta XML-tiedostostaan, jossa oli viitattu mahdollisiin ulkoisiin resursseihin, kuten kuviin, videoihin ja äänitiedostoihin, mikäli artikkeli näitä sisälsi. Kuvassa 5.10 on vielä esitetty esimerkki artikkelin XML-rakenteesta.



Kuva 5.11: Esimerkki artikkelin XML-rakenteesta.

Julkaisuprosessin moottoriksi kehitettiin Ruby-skripti, joka käytti julkaisuun liittyviä staattisia resursseja ja vaihtuvia materiaaleja lähdeaineistona. Lopputuloksena saatiin itsenäinen HTML5-taitettu lehti, jota voitiin käyttää esimerkiksi Baker eBook Framework -ohjelmistokehyksellä toteutetun natiivin sovelluksen tai Stage Framework -ohjelmistokehyksen web-applikaation osana. Lehti toimi myös täysin itsenäisenä kokonaisuutena, joka voitiin julkaista verkkosivuna. Kuvassa 5.12 on esitetty HTML5-taitetun lehden julkaisuun liittyvät resurssit.



Kuva 5.12: Lehden julkaisuprosessiin liittyvät resurssit.

Kuten kuvassa 5.12 esitetään, lehden julkaisemiseen käytettävät materiaalit voidaan jakaa staattisiin ja vaihtuviin. Lehden staattiset materiaalit kattavat käyttöliittymäresurssit ja lehden ulkoasuun ja toiminnallisuuksiin liittyvät resurssit, kuten käytetyt web-kirjasimet sekä LESS-tyylipohjat ja HAML-dokumenttipohjat. Staattiset resurssit siis sisältävät lehden visuaalisen ilmeen, rakenteen ja toiminnallisuuden. Numerokohtaiset, vaihtuvat materiaalit sisältävät artikkelit ja lehteen liittyvän metadatan, kuten sisältösivun XML-formaatissa. Näiden lisäksi artikkelista riippuen materiaalina voi olla kuvatiedostoja, videota tai ääntä. Artikkeleiden kuvista käsiteltiin erikokoiset versiot ennen Ruby-skriptiä perustuen tämän diplomityön ohjaajan kehittämään kuva-algoritmiin [1], joka pyrkii tunnistamaan kuvista tärkeitä kohteita. Algoritmi tunnistaa kuville myös julkaisulle määriteltyjen tehostevärien joukosta lähimmän vastineen, jota käytettiin numerokohtaisena metadatana määrittelemään artikkelikohtainen tehosteväri.

Lehdelle toteutettiin staattisina resursseina myös sovelluksen aloituskuvat ja sovellusikonit, joita voidaan käyttää natiiveissa sovelluksissa tai web-applikaatioissa, esimerkiksi iOS-alustalla, kun web-applikaatio lisätään laitteen kotivalikkoon.

5.3 iOS-sovellus

Aalto University Magazine -demolehdelle toteutettiin iOS-sovellus, joka julkaistiin diplomityön loppuvaiheessa App Storessa ilmaiseksi ladattavaksi[115]. iOS-sovelluksella haluttiin toisaalta demonstroida toteutettua tablettikonseptia, mutta myös konkretisoida HTML5-muodossa julkaistu sisältö osana natiivia sovellusta. Alustaksi olisi voitu valita myös esimerkiksi Android, mutta iOS tarjoaa tällä hetkellä kypsimmän alustan HTML5-taitetun sisällön käyttämiseksi: toisaalta alustan selainmoottorin tuki eri teknologioille on markkinoiden laajinta [15], mutta laitteet tarjoavat myös hyvän suorituskyvyn niin CPU- kuin GPU-intensiiviselle laskennalle [60]. Apple myös pitää vanhemmat laitteet pidempään mukana aktiivisessa ohjelmistotuessa, kuin esimerkiksi suurin kilpailijansa Samsung. iOS-alustan käyttökokemus on siis konsistentimpi kuin millään muulla alustalla tällä hetkellä, mikä tekee siitä ideaalin jakelualustan.

5.3.1 Ohjelmistokehys

iOS-sovelluksen pohjaksi valittiin Baker eBook Framework -ohjelmistokehys, joka on kehitetty nimenomaan digitaalisten julkaisujen, kuten kirjojen ja lehtien paketointiin natiiviksi sovellukseksi. iOS-sovelluksen toteutuksen aikainen Baker-versio ei sisältänyt lehtihyllyä, joten konseptilehti toteutettiin yksittäisenä lehtenä ja yhtenä sovelluksena. Kun käyttäjä käynnistää sovelluksen, avautuu käyttäjälle suoraan demolehden sisältösivu. Mikäli Aalto University Magazine -tablettilehden tuottamista päätettäisiin jatkaa konseptilehden pohjalta, jouduttaisiin sovellukseen toteuttamaan lehtihylly, mikäli käyttäjille haluttaisiin tarjota pääsy aikaisempiin tablettilehden numeroihin. Diplomityön loppuvaiheessa julkaistu Baker 4.0 sisältää tuen iOS:n Newsstand eli Lehtikioski -sovellukselle, joka mahdollistaa eri numeroiden tarjoamisen Newsstand-sovelluksina[116].

Ohjelmistokehyksen natiivin sovelluksen toteutusta muokattiin muun muassa vaihtamalla sivunvaihdon latausikonia ja poistamalla käytöstä yhden kosketuksen siirtymäalueet, joilla käyttäjä voi siirtyä sivulta toiselle napauttamalla sivun reunaan tai edetä tekstissä näkymän korkeuden verran napauttamalla sivun alaosaan. Alueet aiheuttivat jonkin verran vahinkosiirtymiä ja peittivät pienillä näytöillä osan muista painikkeista, kuten kielivalinnasta ja kuvagalleria-ikonista. Ohjelmistokehyksessä otettiin myös käyttöön toimin-

[115] http://itunes.apple.com/fi/app/aalto-university-magazine/id596682061/

[116] http://github.com/Simbul/baker/wiki/4.0-tutorial-for-Newsstand/

nallisuus, jossa julkaisun sisällä esitetyt ulkoisten resurssien, kuten verkkosivustojen linkit avataan sovelluksen sisään avautuvaan selainnäkymään. Toiminnallisuus mahdollistaa verkkosivustojen esikatselemisen sovelluksen sisällä ja nopean paluun takaisin julkaisuun, erotuksena tyypillisestä toimintatavasta, jossa ulkoiset linkit aukaistaan eri sovellukseen, Safari-selaimeen.

5.3.2 Suorituskyky

iOS-sovelluksen suorituskyky on merkittävästi toteutettua web-applikaatiota parempi johtuen pääasiassa lehden paikallisista resursseista, joita ei tarvitse ladata verkon ylitse - sovellus ladataan kerralla laitteeseen. Tulevaisuudessa web-applikaatioiden tilanne voi kuitenkin muuttua oleellisesti, jos alustat alkavat tarjota rajoittamatonta HTML5-sovellusvälimuistia. Tällöin myös web-applikaatioiden resurssit voidaan ladata kokonaisuudessaan laitteen muistiin ensimmäisellä latauskerralla, ja esimerkiksi lehtihyllyn tapauksessa lehtien numeroiden sisällöt voidaan ladata yksi kerrallaan käyttäjän pyytäessä numeroa - yhden numeron resursseja ei kuitenkaan tarvitse ladata reaaliaikaisesti sivu kerrallaan, kuten rajoitetulla HTML5-sovellusvälimuistilla joudutaan tekemään.

5.3.3 Käytettävyysarviointi

iOS-sovelluksen iPad-versiolle suoritettiin käytettävyysarviointi käytettävyyteen keskittyvän Adagen[117] toimesta [61]. Arvioinnin tilaajana toimi Aaltoyliopiston viestintä. Käytettävyystutkimuksen pohjalta toteutettuun sovellukseen ei tehty suuria muutoksia, vaan arvioinnin aineisto jäi Aaltoyliopiston viestinnälle mahdollista tablettilehden käyttöönottoa ja tulevaisuuden kehitystä varten.

Käytettävyysarvioinnissa ei ollut mukana pienemmällä näytöllä varustettua iPhone-laitetta, mikä vaikutti jonkin verran tarjottuihin parannusratkaisuihin: esimerkiksi kainalotekstien sijoittamiseen tarjottiin ratkaisua, jossa kainalotekstit siirretään selkeästi omaksi palstakseen: pienemmillä näytöillä tämä ei ole kuitenkaan mahdollista. Ohjeita voitaisiin kuitenkin soveltaa arvioinnissakin käytetylle suuremman resoluution laitteille.

[117] `http://adage.fi/`

5.4 Web-applikaatio

iOS-sovelluksen lisäksi diplomityössä toteutettiin web-applikaatio [62], jonka tavoitteena oli mahdollistaa lehtien lukeminen alustariippumattomasti ja natiiveista sovelluksista tutulla pyyhkäisyeleellä. Web-applikaatiolle toteutettiin lehtihylly, jotta sen avulla voitaisiin jaella useita numeroita samasta lehdestä. Diplomityön aikana web-applikaation ympärille kehittyi ohjelmistokehys, Stage Framework, joka tarjoaa julkaisijoille mahdollisuuden käyttää web-applikaatiota omien HTML5-muotoisten lehtien tai muun sisällön jakelemiseen.

5.4.1 Lehtihylly

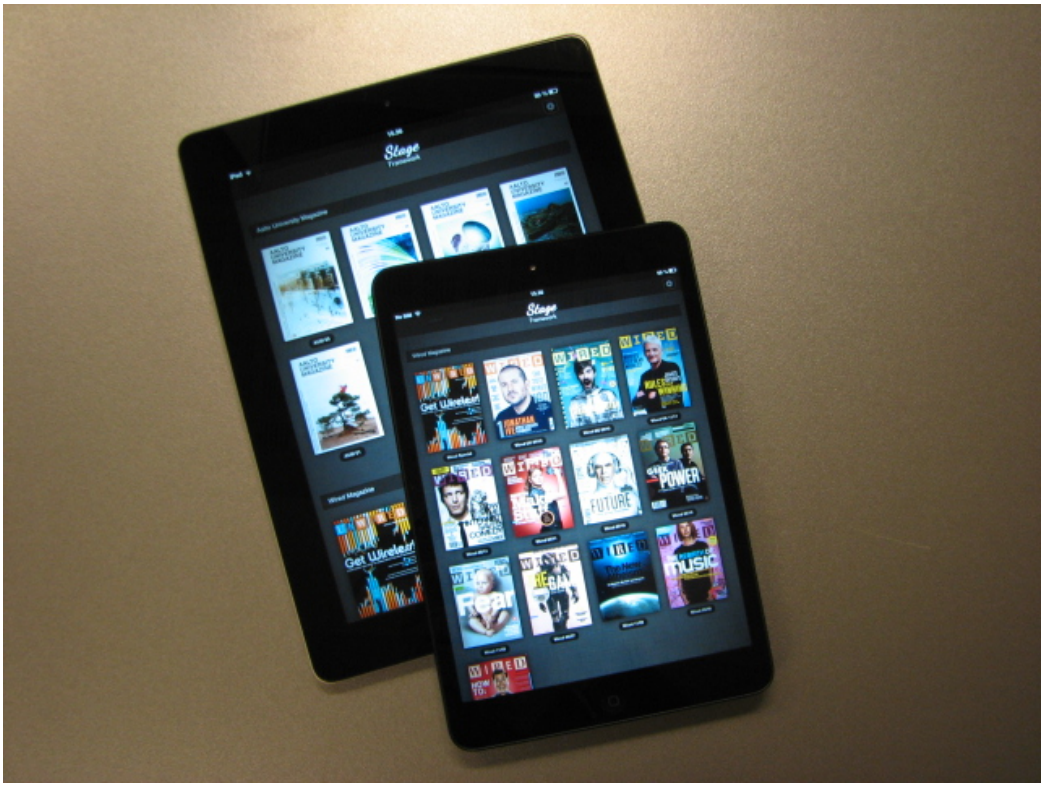
Lehtihylly toteutettiin responsiivisesti käyttäen kolmea kansikuvatarkkuutta ja kahta kansikuvien asettelua. Tableteissa ja suuremmilla resoluutioilla numeroita esitetään neljä rinnan, kun taas pienemmillä resoluutioilla numeroita mahtuu riville kolme. Kansikuvien tarkkuudet on jaettu alle 1024:n ja 2048:n pikselin leveydelle sopiviin ja alkuperäiseen kuvatarkkuuteen, jota käytetään yli 2048 pikseliä ylittävillä resoluutioleveyksillä. Kolmen kuvatarkkuuden tarjoaminen vähentää lataustarvetta tabletti- ja mobiililaitteilla ja toisaalta pikselitiheydeltään suuremmat näytöt saavat tarkemmat kuvat käytettäväksi. Kuvassa 5.13 on esitetty lehtihylly iOS-simulaattorissa iPhone-laitteella, kun kuvassa 5.14 nähdään lehtihylly toiminnassa iPad 3 ja iPad Mini -laitteilla.

Kuva 5.13: Stage Framework -ohjelmistokehyksen web-applikaation lehtihylly iPhone-laitteessa.

Lehtihyllyn tiedot haetaan palvelimelta JSON-tiedostona ja se voidaan tarvittaessa ladata HTML5-sovellusvälimuistiin muiden web-applikaation käyttöliittymän resurssien kanssa. Tällöin lehtihylly on selattavissa offline-tilassa ensimmäisen latauksen jälkeen. Web-applikaatioon olisi ollut mahdollista toteuttaa myös esimerkiksi numeroiden lataaminen offline-tilaan, mutta esimerkiksi Aalto University Magazine -demolehden verrattain suuri koko esti vielä toistaiseksi HTML5-sovellusvälimuistin hyödyntämisen tähän tarkoitukseen. Sovellusvälimuistien tallennustilamäärien kasvaminen eri selaimissa mahdollistaa kuitenkin tulevaisuudessa kyseisen toiminnallisuuden toteuttamisen.

Lehtihylly tukee useiden eri julkaisujen tarjoamista yhdessä, jolloin eri julkaisut erotetaan toisistaan otsake-elementeillä. Julkaisujen ja sen numeroiden järjestys voidaan määritellä palvelimelta ladattavassa JSON-tiedostossa. Lehtihyllyn sisältö ladataan sovelluksen lataamisen jälkeen erillisellä XHR-pyynnöllä, jonka jälkeen lehtihyllyn elementit generoidaan saadun vastauksen

perusteella. Muun muassa iOS-alustalla web-applikaatio ei voi pitää lehtihyllyn sisältöä muistissa lehden numeroita selattaessa, joten se joudutaan generoimaan joka kerta uudestaan, kun selausnäkymästä palataan lehtihyllyyn.



Kuva 5.14: Lehtihylly iPad 3 ja iPad Mini -laitteissa sisältäen Aalto University Magazine -lehtien lisäksi muiden lehtien kansia demonstraatiotarkoituksessa.

Lehtihylly avautuu ensimmäiseksi näkymäksi web-applikaation latautumisen jälkeen ja käyttäjä voi siirtyä lehden selausnäkymään vierittämällä ja valitsemalla lehtihyllystä jonkin lehden numeron. Lehtihyllyn sisällön sisältävän JSON-tiedoston mukana web-applikaatioon ladataan tieto jokaisen numeron sisältämistä HTML-resursseista eli lehden sivuista. Näiden resurssien perusteella web-applikaatio päättelee numeron tarvitsemien sivuelementtien määrän ja lataa ensimmäisenä listatun resurssin numeron etusivuna. Web-applikaatio pitää kirjaa viimeksi selatusta sivusta numerokohtaisesti ja tallentaa tiedon HTML5-tallennustilaan. Jokaisen lehden numeron selaamista jatketaan siis aina siitä kohdasta mihin edellisellä kerralla on jääty.

5.4.2 Selausnäkymä

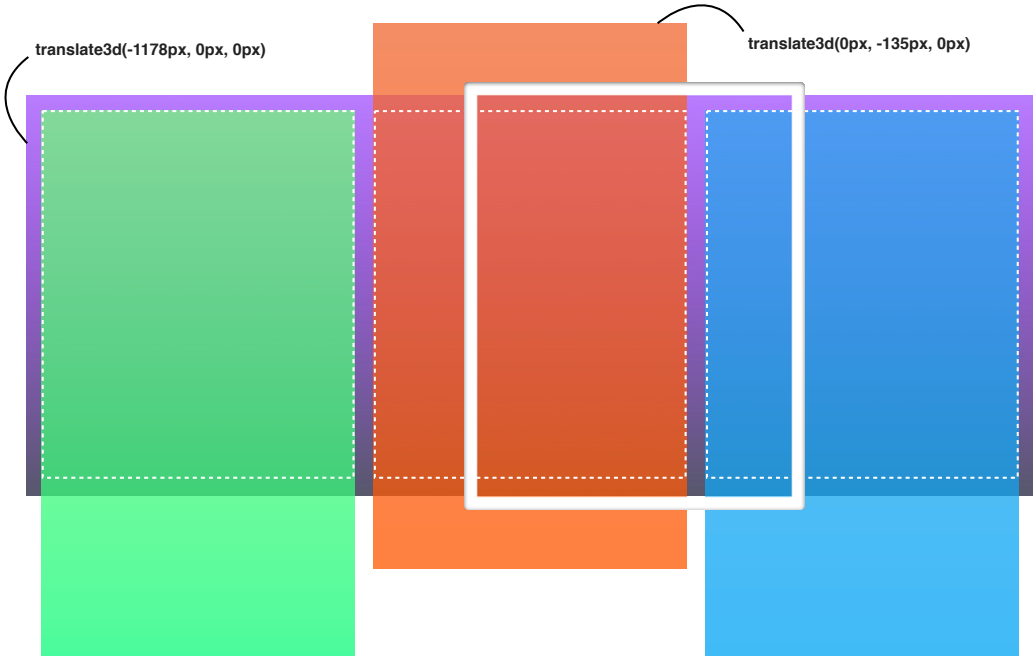
Lehden selausnäkymä koostuu rinnakkain asetelluista sivuelementeistä, jotka generoidaan numeron valinnan jälkeen web-applikaation saamien tietojen perusteella. Jokaiseen sivuelementtiin ladataan sille määritelty HTML-resurssi joko sivulle siirryttäessä tai mahdollisesti etukäteen eri välimuistitustekniikoita käytettäessä. Käyttäjä pystyy liikkumaan eri sivujen välillä pyyhkäisemällä, jolloin sivuelementtien isäntäelementtiä liikutetaan suhteessa tämän isäntäelementtiin käyttäen kosketuksen aikana CSS3-transformaatiota ja kosketuksen jälkeen CSS3-siirtymiä. Numeron sisällä voidaan käyttää myös sisäisiä linkkejä, jotka viittaavat julkaisussa määriteltyihin HTML-resursseihin eli lehden sivuihin. Sisäiset linkit tunnistetaan ja erotetaan ulkoisista HTML-resursseista web-applikaatiossa JavaScriptin avulla. Kun käyttäjä painaa sisäistä linkkiä, siirrytään selausnäkymässä kyseiselle sivulle.

Selausnäkymässä generoidaan sivukohtaisesti latausanimaatioita käyttäen CSS3-tekniikkaa. Kehityksen aikana CSS3-animaatioilla toteutetut latausanimaatiot osoittautuivat raskasta DOM-manipulaatiota hyvin kestäviksi - ne siis toimivat sulavammin raskaiden DOM-operaatioiden, kuten julkaisun sivun lataamisen aikana, kuin esimerkiksi GIF-kuvat, JavaScript-animoinnit tai jopa HTML5-piirtoalueella toteutetut animaatiot. Sivuilla, joita ei ole vielä kertaakaan näytetty käyttäjälle, on oma esirippu-elementti, joka animoidaan läpinäkyväksi CSS3-siirtymällä ja poistetaan sivun edestä sivulle saavuttaessa. Tämä johtuu sivun sisällön satunnaisista välähdyksistä, kun sille asetetaan ensimmäisen kerran CSS3-transformaatioasetus sivun vieritystoiminnallisuuden valmistelemiseksi. Mikäli esivalmistelua ei tehtäisi, välähtäisi sisältö käyttäjän vierittäessä yksittäistä sivua ensimmäisen kerran.

Käyttäjät voivat siis vierittää selausnäkymän sivuja, mikäli sisältö ei mahdu selainnäkymään. Vierityksen toteuttamiselle oli useita ratkaisuvaihtoehtoja, joita käsitellään tarkemmin seuraavassa alaluvussa. Lopullinen vieritystoiminnallisuus perustui kuitenkin CSS3-transformaatioiden käyttöön kosketuksen aikana ja JavaScript-silmukalla suoritettavaan transformaatioon kosketuksen loputtua. Selausnäkymän sivuelementtejä voi vierittää tai pyyhkäistä yli, jolloin sivu kohdennetaan kimpoamisella (engl. bounce back).

Kuvassa 5.15 on esitetty selausnäkymän rakenne. Valkoisella katkoviivalla merkityt elementit esittävät yksittäisten sivujen isäntäelementtejä, jotka on sijoitettu vaakatasossa vierekkäin violetilla merkityn elementin sisään. Violetin elementin isäntäelementti on kiinteäreunainen valkoinen kehys,

joka myös esittää käyttäjän näkemää aluetta. Sivujen väliin on jätetty marginaalia kuvan selventämiseksi. Kuvassa käyttäjä on vierittänyt oranssilla sävyllä merkittyä sivua, joka on vaikuttanut sivun sisällön sisältävän elementin *transform*-asetukseen. Käyttäjä on kuitenkin myös aloittanut siirtymisen oranssilta sivulta siniselle käyttämällä pyyhkäisyä, mikä vaikuttaa reaaliaikaisesti violetin elementin *transform*-asetukseen vähentämällä *translate3d*:n x-akselin arvoa. Kuvassa 5.16 on esitetty kuvattu tilanne laitteen näytöllä.



Kuva 5.15: Lehden selausnäkymän rakenne.

Kuva 5.16: Lehden selausnäkymän rakenteen yhteydessä kuvattu tilanne laitteen näytöllä.

5.4.2.1 Pyyhkäisy ja kineettinen vieritys

Web-applikaation selausnäkymän vaatimuksena oli sivujen välillä siirtymisen mahdollistaminen pyyhkäisyelettä käyttämällä. Siirtymän toteuttamiselle oli alunperin olemassa kolme vaihtoehtoa: selaimen vaakatason vieritys, DOM-elementtien JavaScript-manipulaatio sekä CSS3-transformaatio ja -siirtymät. Selaimen vaakatasossa tapahtuvan vierityksen käyttäminen ei kuitenkaan ollut käytännöllinen vaihtoehto, sillä selaimien vieritystoteutukset vaihtelevat suuresti. Lisäksi ongelmia olisi aiheuttanut sisällön pituuden vaihtelu eri sivujen välillä. Selaimen vieritystä käyttämällä ei myöskään ollut mahdollista saavuttaa luonnollisesti toimivaa snapping-toiminnallisuutta sivujen kohdentamiselle vierityksen päätyttyä.

Toinen vaihtoehto selausnäkymän liikuttamiselle vaakasuunnassa oli JavaScriptillä tapahtuva elementtien sijainnin manipulointi ja animointi. Sivujen sisällön sisältäviä elementtejä voidaan manipuloida yksi kerrallaan tai yhdessä isäntäelementin kanssa. Ongelmana pelkkään JavaScript-ratkaisuun tukeutumisessa on kuitenkin HTML-sisällön piirron raskaus, erityisesti paljon elementtejä, kuten kuvia sisältävällä julkaisulla. Tästä syystä pyyhkäisyn toteuttamisessa päädyttiin kolmanteen ratkaisuun eli CSS3-transformaatioon ja -siirtymiin.

CSS3-ratkaisun etuna on rautapohjaisen kiihdytyksen hyödyntäminen sisällön liikuttamisessa. Pyyhkäisyä ei kuitenkaan voida toteuttaa ilman JavaScriptiä, sillä jo DOM-kosketustapahtumien eli käyttäjän kosketuksien hyödyntäminen vaatii JavaScript-toiminnallisuutta. Web-applikaation selausnäkymän toteuttamisessa käytettiin pohjana Swipe.js-kirjastoa, josta kerrottiin aikaisemmin luvussa 3.5.4. Swipe.js toteuttaa pyyhkäisyssä käytettävän sivujen kohdentamisen, kun käyttäjä vapauttaa kosketuksen. Kirjastossa on myös toteutettu ensimmäisen ja viimeisen sivuelementin ylipyyhkäisyssä tapahtuva sisällön reunasta aiheutuva resistanssi, jossa sisällön liikuttaminen kosketuksella hidastuu mitä kaueammaksi sisällön reunasta yritetään pyyhkäistä.

Pyyhkäisyn toteuttamiseksi CSS3-tekniikoita joudutaan käyttämään kahdessa osassa: ensin CSS3-transformaatiota kosketuksen aikana ja CSS3-siirtymiä kosketuksen loppuessa. Kosketuksen aikana elementtien sijaintia muutetaan reaaliaikaisesti jokaisen DOM-kosketustapahtuman eli pääasiassa *touchmove*-tapahtuman aikana riippuen kosketuksen vaakasuuntaisesta liikkeestä. Kun kosketus päättyy *touchend*-tapahtumaan, lopetetaan CSS3-transformaatiolla tapahtuva sijainnin päivittäminen ja asetetaan sivu kohdentumaan selaimen näköalueeseen (engl. viewport) käyttäen CSS3-siirtymää, joka suorittaa animoinnin automaattisesti ja rautapohjaista kiihdytystä hyödyntäen.

Web-applikaation selausnäkymään toteutettiin myös sivukohtainen kineettinen vieritys [63], joka on tuttu alunperin iOS-alustalta. Kineettisessä vierityksessä vierityksen liike-energia pienenee tasaisesti kosketuksen loputtua, kunnes sisältö pysähtyy. Vierityksen toteutuksessa käytettiin sivunvaihdosta tuttua CSS3-ratkaisua, mutta kosketuksen loputtua vierityksen hidastumista ei toteuteta CSS-siirtymillä vaan edelleen CSS-transformaatiolla. Jokaisen sijainnin päivityksen jälkeen jatketaan JavaScript-silmukan suorittamista, joka määrittelee elementin seuraavan sijainnin eli toteuttaa kineettisen vierityksen puoliintumisen ajan suhteen. JavaScript-silmukka asettaa suorituskyvyllisiä haasteita web-applikaation selausnäkymän toiminnalle, joten toteutukselle kokeiltiin myös vaihtoehtoisia ratkaisuja, kuten CSS-animaatioita käyttäen kosketuksen loputtua laskettavia avainkehyksiin (engl. keyframes) perustuvia sijainteja. Muilla toteutuksilla ei kuitenkaan onnistuttu toteuttamaan yhtä luotettavasti toimivaa vieritysratkaisua.

CSS-transformaatioon perustuvan vierityksen sijasta selausnäkymässä olisi voitu käyttää selaimen luonnollista vieritystä käyttämällä *-webkit-overflow-scrolling: touch* -asetusta, joka mahdollistaa ylitsevuotavien (engl. overflow) elementtien vierittämisen kosketuksella tietyillä mobiiliselaimilla. Asetuksen tuki on kuitenkin erittäin rajallista, joten se ei olisi ollut alustariippumaton ratkaisu. Erityisesti Android Browser -selaimen vanhemmat versiot ovat olleet ongelmallisia muiden kuin kokonaisten sivujen vierittämisen mahdollistamisessa[118].

5.4.2.2 Sisällönhallinta ja suorituskyky

Web-applikaation sisällönhallinnassa jouduttiin pohtimaan täysin uudenlaisia toteutusratkaisuja, sillä itsenäisten HTML-resurssien käyttäminen web-applikaation sisällä asettaa ne osaksi web-applikaatiota. Tämä vaikuttaa muun muassa sisällön tyyleihin ja resurssien sijainteihin. Web-applikaatiolle toteutettiin kaksi sisällönhallintamoottoria, aluksi kokeellinen *iframe*-elementtiin perustuva moottori, joka lataa ulkoiset HTML-resurssit omaan hiekkalaatikkoonsa (engl. sandbox), jossa resurssi pidetään erillään web-applikaatiosta ja sen DOM-rakenteesta. Kehitetyllä moottorilla voidaan esittää julkaisuja, jotka eivät tarvitse interaktiivisuutta, kuten julkaisun sisäisiä linkkejä, vaan selviävät ainoastaan web-applikaation tarjoamalla pyyhkäisyyn perustuvalla sivunvaihdolla. Mielenkiintoisesti kuitenkin esimerkiksi HTML5:n *video*-elementillä toteutettu video on toistettavissa myös tällä moottorilla.

[118] `http://barrow.io/overflow-scrolling/`

Varsinainen kehitetty sisältömoottori perustui LESS-tekniikan käyttöön. Web-applikaatioon ladattava itsenäinen HTML-dokumentti ja sen resurssit muokataan sopivaksi toisen sovelluksen osaksi. Tämä vaatii muun muassa ladattavan resurssien tyylien istuttamista osaksi web-applikaatiota ja sen DOM-rakennetta. Esimerkiksi *html*- ja *body*-elementit joudutaan muuttamaan *html*- ja *body*-luokiksi niin web-applikaatioon tuotavassa HTML-dokumentissa kuin sen kaikissa CSS-tyyliresursseissa. Kaikille tyylisäännöille lisätään myös tuotavan HTML-sisällön säiliöelementille (engl. container) asetettu *id*-attribuutti, joka asettaa kaikki tyylit web-applikaation kontekstiin. Kontekstin vaihtuminen asettaa vaatimuksia HTML-resurssien tyyleille: esimerkiksi rem-yksikköä ei voida käyttää web-applikaatioon tuotavassa sisällössä, sillä sisällön juurielementti (engl. root element) vaihtuu alkuperäisestä web-applikaation juurielementtiin. Tämä vaikuttaa rem-yksiköllä asetettuihin sääntöihin, kuten tyypillisesti kirjasinten kokoon.

Tyylien manipuloinnissa käytetään JavaScriptillä suoritettavaa LESS-kirjastoa, jolla voidaan muun muassa asettaa tyyleissä käytetyille resursseille, kuten taustakuville, uusia polkuja suhteessa web-applikaatioon. Web-applikaatioon tuotavista HTML-resursseista parsitaan kaikki käytetyt JavaScript-resurssit, jotka suoritetaan joka kerta kun sivu näytetään käyttäjälle web-applikaation selausnäkymässä. Tämä mahdollistaa muun muassa JavaScriptiin perustuvien animaatioiden suorittamisen joka kerta kun sivu näytetään. Käytäntö on sama kuin toteutetussa iOS-sovelluksessa käytetyssä Baker eBook Framework -ohjelmistokehyksessä.

Web-applikaation sisällönhallinta on erittäin tarkkaa suorituskyvyn kannalta, mutta myös koko web-applikaation vakauden kannalta. Uuden sisällön tuominen web-applikaatioon toteutetun sovelluksen mittakaavassa on raskasta, sillä web-applikaation DOM-rakenteen osaksi joudutaan tuomaan kokonaisia HTML-resursseja ja niille joudutaan suorittamaan verrattain raskasta käsittelyä reaaliaikaisesti. Erityisesti iOS-alustalla web-applikaation vakaus riippuu selaimen muistiin jätettävistä resursseista, kuten taustakuvista, tai jopa elementtien taustaväreistä. Web-applikaatioon tuotavaa lehtien sisältöä ei voida pitää sovelluksen muistissa ladattaessa uutta, sillä web-applikaatio kaatuu hyvin nopeasti. Esimerkiksi vanhat sivut on poistettava DOM-rakenteesta, mutta tätä ennen jokaisen kuvan, videon ja muun resurssin *src*-attribuutit tai CSS:llä asetetut taustakuvat on poistettava. Viittausten jättäminen ennen DOM-rakenteesta poistamista johtaa ennen pitkää web-applikaation eli selaimen kaatumiseen erityisesti mobiilikäyttöjärjestelmillä. Käytetyssä CSS-transformaatioon perustuvassa selausnäkymässä myös elementtien taustavärin tyhjentäminen ennen poistoa osoittautui välttämättömäksi - syynä voi olla esimerkiksi elementtien taustaväristä tuotettavien

tekstuurien jääminen selaimen muistiin, sillä transformaatiossa tukeudutaan GPU-pohjaiseen kiihdytykseen. Sisältöä joudutaan siis lataamaan jatkuvasti uudestaan, vaikka se olisi jo kertaalleen ladattu. Suorituskyvyn parantamiselle on kuitenkin olemassa erinäisiä toimenpiteitä, joista diplomityön aikana kehitetystä dynaamisesta HTML5-sovellusvälimuistista kerrotaan seuraavassa.

5.4.2.3 Dynaaminen HTML5-sovellusvälimuisti

Web-applikaation sisällönhallinnan tueksi kehitettiin dynaaminen HTML5-sovellusvälimuisti, joka mahdollistaa sisällön välimuistittamisen eri tavoilla ja siten nopeuttaa web-applikaatiolla tapahtuvaa julkaisujen selaamista. Dynaaminen HTML5-sovellusvälimuisti pyrkii helpottamaan sisällön toistuvasta lataamisesta aiheutuvaa suorituskykyrasitetta, joka juontaa juurensa aikaisemmin kuvatusta web-applikaation käytettävissä olevan muistin rajallisuudesta. Tavallisesti HTML5-sovellusvälimuistia käytetään staattisesti lataamalla manifest-tiedosto ja sovellusvälimuistin sisältö web-applikaation latautuessa tai mahdollisesti päivittämällä sovellusvälimuistin sisältö ulkoisten tekijöiden johdosta, kuten lehtihyllyn sisällön päivittyessä tai lehden numeron sisällön muuttuessa.

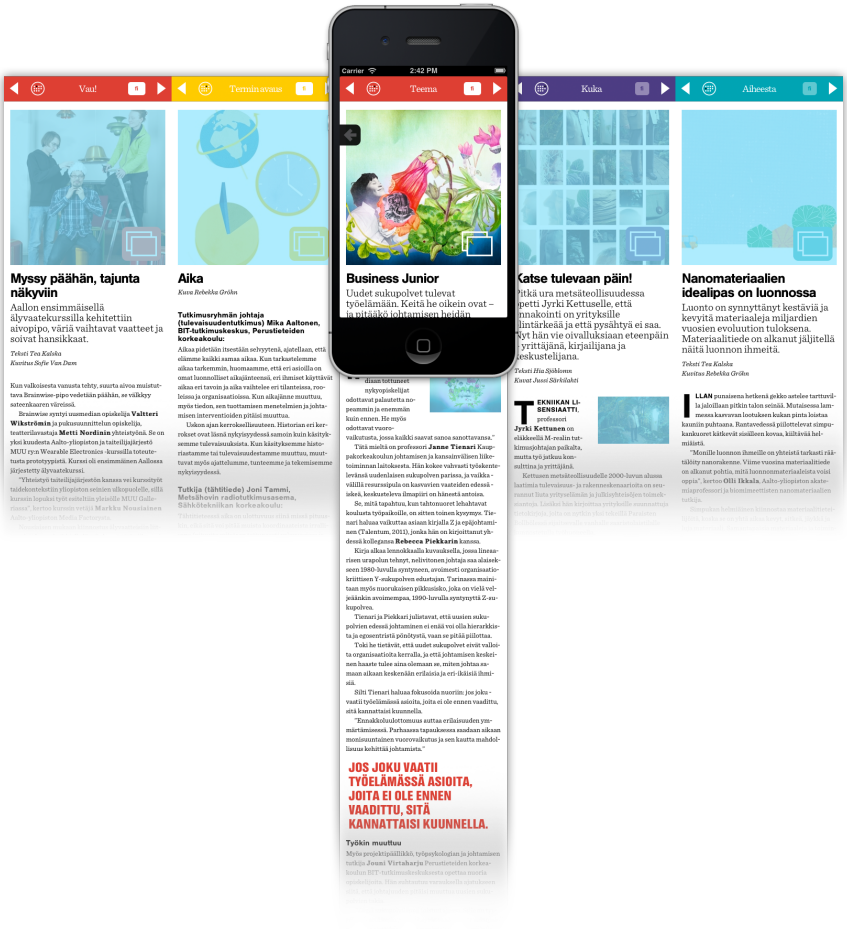
Tämän diplomityön luvussa 3.3 nostettiin esiin Paul Irishin parhaat käytännöt, joissa mainittiin HTML5-sovellusvälimuistin hyödyntäminen myös online-sovelluksissa. Diplomityössä kehitetty dynaamisen HTML5-sovellusvälimuistin konsepti vie tämän käytännön vielä yhden askeleen pidemmälle. Sovellusvälimuistia voidaan muokata reaaliaikaisesti jokaisen asiakaslaitteen toimesta, jolloin sovellusvälimuistin sisältöä voidaan mukauttaa kunkin käyttäjän toiminnan perusteella ja tarjota se yksilöllisesti asiakaslaitteelle, myös pitkän käyttökatkon jälkeen. Dynaamisen HTML5-sovellusvälimuistin lopullisessa toteutuksessa Stage Framework -ohjelmistokehyksen osana tukeudutaan sovellusvälimuistin lisäksi HTML5-tallennustilaan, perinteisiin evästeisiin sekä palvelinpuolella PHP:hen ja SQLite-tietokantaan.

Dynaamisessa HTML5-sovellusvälimuistissa jokaiselle laitteen ja selaimen yhdistelmälle generoidaan uniikki tunniste, joka tallennetaan sekä laitteen HTML5-tallennustilaan että palvelimen SQLite-tietokantaan. SQLite-tietokannassa pidetään kirjaa tunnisteista ja niihin liittyvistä HTML5-sovellusvälimuistiin tallennettavista resursseista. Esimerkiksi web-applikaation käyttöliittymä voi lisätä, poistaa tai tyhjentää palvelimelle tallennetut resurssiviittaukset kyseisen laitteen ja selaimen tunnisteelle. Tyypil-

linen käyttötapaus voisi koostua tilanteesta, jossa käyttäjä valitsee web-applikaatiossa offline-tilaan ladattavat lehtien numerot. Web-applikaation käyttöliittymästä lähetetään POST-pyyntö palvelimelle sisältäen tiedot välimuistiin haluttavista resursseista. Varsinainen resurssien listaaminen voidaan toteuttaa joko käyttöliittymässä tai palvelimella esimerkiksi lehden numeroiden tunnisteen perusteella.

Kun palvelin tallentaa tiedot laitteen ja selaimen tunnisteelle välimuistitettavista resursseista käyttäen SQLite-tietokantaa, voidaan asiakaslaitteella pyytää HTML5-sovellusvälimuistin päivittämistä esimerkiksi JavaScriptin avulla. Palvelimelta pyydettävä manifest-tiedosto generoidaan reaaliaikaisesti esimerkiksi PHP:n avulla. Kehitetyn dynaamisen HTML5-sovellusvälimuistin toteutuksessa PHP voi lukea asiakaslaitteen ja -selaimen tunnisteen suoraan evästeestä, johon se on tallennettu web-applikaation toimesta. Välimuistitettavat resurssit haetaan SQLite-tietokannasta käyttäen tunnistetta ja lähetetään web-applikaatiolle osana manifest-tiedostoa, joka voi sisältää myös muita esimerkiksi web-applikaation offline-toiminnan mahdollistavia resursseja.

Kuvissa 5.17 ja 5.18 on esitetty kaksi mahdollisuutta dynaamisen HTML5-sovellusvälimuistin hyödyntämiseen Stage Framework -ohjelmistokehyksen tapaisen web-applikaation osana. HTML5-sovellusvälimuistiin tallennettavan tiedon määrä on toistaiseksi rajallinen eri alustoilla, joten esimerkiksi toteutetun demolehden koko sisällön lataaminen HTML5-sovellusvälimuistiin ei ole aina mahdollista. Käytettäviä yhden numeron resursseja voidaan kuitenkin ladata HTML5-sovellusvälimuistiin osissa, esimerkiksi sivuryhmissä tai kuvissa 5.17 ja 5.18 esitetyllä tavalla. Kuvassa 5.17 havainnollistetussa toimintatavassa välimuistiin ladataan yhden numeron tapauksessa kaikki kuvaresurssit, jotka muodostavat suuren osan siirrettävästä datamäärästä. Tämä nopeuttaa oleellisesti lehden selauskokemusta erityisesti pienikaistaisilla yhteyksillä. Kuvassa 5.18 on esitetty vaihtoehtoinen tapa, jossa ladataan lehden kaikki muut resurssit, kuten CSS-tyylimäärittelyt, HTML- ja JavaScript-resurssit. Näin toimimalla voidaan panostaa suoritettavien HTTP-kutsumäärien rajoittamiseen.



Kuva 5.17: Lehden sisällöstä voidaan välimuistittaa esimerkiksi suurimmat resurssit, kuten artikkeleiden kuvat.

Kuva 5.18: Lehden sisällöstä voidaan välimuistittaa myös kaikki muut resurssit, poislukien raskaat kuvaresurssit, jolloin minimoidaan HTTP-kutsujen määrää.

5.4.3 Alustariippumattomuus

Toteutettu web-applikaatio on lähtökohtaisesti alustariippumaton ja pyrkii hyödyntämään diplomityössä kuvattuja web-standardeja. Suurimmaksi alustariippumattomuutta rajoittaviksi tekijöiksi muodostuivat kuitenkin web-standardeja tukemattomat selaimet sekä tabletti- ja mobiililaitteiden vaihteleva suorituskyky uusien teknologioiden hyödyntämisessä. Alusta- ja lai-

tekohtaisia suorituskykyeroja on kartoitettu toteutetuissa suorituskykymittauksissa, joita esitellään myöhemmin luvussa 6.2.

Parhaimmaksi alustaksi web-applikaation toiminnalle osoittautui iOS, jolla web-applikaatio myös integroituu parhaiten natiivien sovellusten joukkoon kotivalikkoon lisäämisellä. Web-applikaatiolle määriteltiin aloituskuvat ja sovellusikonit, jotka ovat myös muiden alustojen käytettävissä. Web-applikaation toimivuudessa seuraavaksi parhaaksi alustaksi osoittautui Android, jonka oletusselain Android Browser käyttää samaa selainmoottoria iOS-alustan selaimien kanssa. Epävarmimmin alustariippumattomuutta painotettaessa web-applikaatio suoritui Windows Phone -alustoilla, joissa ongelmia aiheuttivat DOM-kosketustapahtumien hyödyntäminen ja selaimen renderöintimoottorin käyttäytyminen. Esimerkiksi *fixed*-arvon käyttäminen CSS-tyyleissä *position*-ominaisuudelle aiheutti ongelmia sivua vieritettäessä. [62]

5.4.4 Ohjelmistokehys

Toteutetun web-applikaation ympärille kehitettiin diplomityön aikana ohjelmistokehys, Stage Framework, joka julkaistiin myöhemmin avoimesti saataville. Ohjelmistokehyksen tarkoituksena on tarjota tarvittavat resurssit HTML5-taitetun sisällön julkaisemiselle yleisimmille tabletti- ja mobiilialustoille käyttäen kehitettyä web-applikaatiota. Ohjelmistokehyksen sisältö ja käyttö esitellään tarkemmin tulokset-luvun alaluvussa 6.3.

Tulokset-luvussa esitellyt alusta- ja selainkartoitus, sekä suorituskykymittaukset on toteutettu silmällä pitäen web-applikaatiossa ja siten ohjelmistokehyksessä käytettyjä käyttöliittymäteknologioita. Kuten tässä alaluvussa 5.4 on esitelty, on ohjelmistokehyksen käyttöliittymän toiminnassa yhtymäkohtia aikaisemmin luvussa 4 esiteltyjen referenssijulkaisujen kanssa, vaikka jokainen julkaisuista on toteutettu viime kädessä täysin erilaisella tekniikalla. Ohjelmistokehyksen suunnittelussa on pyritty identifioimaan ja tarjoamaan parhaat mahdolliset käytännöt HTML5-muotoiselle digitaaliselle julkaisemiselle tämänhetkisessä toimintaympäristössä.

Luku 6

Tulokset

Tässä luvussa käsitellään diplomityön tuloksia kuten alusta- ja selainkartoitusta, suorituskykymittauksia ja Stage Framework -ohjelmistokehystä. Digitaaliseen julkaisemiseen HTML5-tekniikalla liittyviä ohjenuoria käsitellään myöhemmin luvussa 7. Erityisesti suorituskykymittauksista saatuja tuloksia ei ole voitu niiden laajuuden vuoksi esittää kokonaisuudessaan tässä tuloksetluvussa, joten ne on julkaistu avoimesti saataville [60], sekä ovat saatavilla digitaalisesti pyynnöstä. Myös alusta- ja selainkartoitus on julkaistu saataville erillisenä dokumenttina [15].

6.1 Alusta- ja selainkartoitus

Alusta- ja selainkartoituksella haluttiin selvitää kuinka hyvin eri alustat ja selaimet tukevat diplomityön sovelluksissa käytettyjä teknologioita. Erityisen kiinnostavia olivat toteutetun web-applikaation osana käytetyt teknologiat, kuten HTML5-sovellusvälimuisti ja CSS3-transformaatio, sekä sisällön puolesta tuetut HTML5:n video- ja audio-formaatit, sekä webkirjasinten tila. Alusta- ja selainkartoitus suoritettiin samalla itse toteutetulla verkkosovelluksella suorituskykymittausten kanssa. Eri teknologioiden tuen selvittämiseen käytettiin kohdassa 3.5.2 esiteltyä Modernizr-kirjastoa, joka sisältää kattavan tuen myös uusille teknologioille. Tämän alaluvun osassa 6.1.3 on esitelty erikseen CSS3-kirjasinmoduulin kartoitus ja sen tulokset spesifikaation tuoreuden ja vähäisen tuen, mutta myös digitaaliselle julkaisemiselle tarjoavan erityisen mielenkiintoisuutensa vuoksi.

6.1.1 Kartoitetut teknologiat

Alusta- ja selainkartoitukseen valitut teknologiat voidaan jakaa seitsemään pääkategoriaan, jotka on esitetty seuraavassa taulukossa 6.1.

Taulukko 6.1: Kartoitetut selainteknologiat alusta- ja selainkartoituksessa.

Kategoria	**Teknologia tai ominaisuus**	**Alkuperäinen nimi (jos suomenn.)**
HTML5-tekniikat	HTML5-sovellusvälimuisti	HTML5 Application Cache
	Web-soketit	Web Sockets
	HTML5-tallennustila, paikallinen	Web Storage, Local
	HTML5-tallennustila, sessio	Web Storage, Session
	Taustasuorittajat	Web Workers
	HTML5-audio-elementti	HTML5 Audio Element
	HTML5-piirtoalue	HTML5 Canvas Element
	HTML5-video-elementti	HTML5 Video Element
	WebGL-rajapinta	WebGL API
CSS3-tekniikat	CSS-tavutus	CSS Hyphenation
	Web-kirjasimet	Web Fonts
	CSS3-transformaatiot	CSS3 Transforms
	CSS3-3D-transformaatiot	CSS3 3D Transforms
	CSS3-animaatiot	CSS3 Animations
	CSS3-siirtymät	CSS3 Transitions
	CSS3-kulmapyöristys	CSS3 border-radius
	CSS3-varjot	CSS3 box-shadow
	CSS3-tekstivarjot	CSS3 text-shadow
	CSS3-taustakuvakoko	CSS3 background-size
	CSS3-monitaustakuva	CSS3 multiple backgrounds
Web-kirjasinformaatit	EOT	
	TTF/OTF	
	SVG	
	WOFF	
Video-formaatit	H.264/MPEG-4 AVC	
	Ogg/Theora	
	WebM/VP8	
Audio-formaatit	M4A	
	MP3	
	Ogg	
	WAV	
Mediakyselyt	width	
	height	
	device-width	
	device-height	
	orientation	
	aspect-ratio	

	device-aspect-ratio	
	device-pixel-ratio	
SVG-tekniikat	SVG	SVG
	SVG CSS-taustakuvissa	SVG in CSS backgrounds
	SVG HTML-kuvaelementissä	SVG in HTML img element
	SVG HTML5:n seassa	SVG inline with HTML5
Muut tekniikat	DOM-kosketustapahtumat	DOM Touch Events
	JSON (JavaScrip-objektinotaatio)	JSON (JavaScript Object Notation)

HTML5-tekniikat -kategorialla haluttiin kartoittaa suurimmat uudet teknologiat ja selvittää kuinka suuren tallennustilan selaimet tarjoavat HTML5:n uudelle sovellusvälimuistille ja tallennustilalle. HTML5-sovellusvälimuistin ylärajan testaaminen laitteilla, jotka eivät aseta ylärajaa oli kuitenkin käytännössä mahdotonta, joten suurimmaksi testattavaksi tilamääräksi asetettiin 101 megatavua, joka riittää hyvin digitaalisessa julkaisemissa esimerkiksi lehden numeroiden tallentamiseen laitteeseen offline-lukemista varten. Hyvin monet tuloksissa 101 megatavua tallentaneet selaimet tarjoavatkin teoriassa rajattoman HTML5-sovellusvälimuistin, mutta käytännössä näilläkin selaimilla on rajansa - erityisesti mobiililaitteissa. HTML5-tallennustilan osalta testattiin erikseen Local Storage ja Session Storage -tallennustilat. HTML5-tekniikat -kategoriassa testattiin myös tuet audio-, piirtoalue- ja video-elementeille, sekä reaaliaikaisen 3D-grafiikan selaimessa mahdollistavan WebGL-rajapinnan tuki.

CSS3-tekniikat -kategoriassa testattiin CSS-tavutus ja web-kirjasinten tuki yleisesti. Eri web-kirjasinformaatit testattiin jokainen erikseen visuaalisella varmistuksella. Suuremmista CSS-teknologioista testattiin 2D- ja 3D-transformaatiot, sekä animaatiot ja siirtymät. Pienempinä teknologioina kuriositeetiksi mukaan otettiin elementtien kulmien pyöristys ja varjostus, sekä tekstien varjostus. Lisäksi testattiin taustakuvan koon säätäminen uusilla asetuksilla, sekä useamman taustakuvan tuki. Valittuja pienempiä CSS:n ominaisuuksia on käytetty toteutetussa Stage Framework -ohjelmistokehyksen web-applikaatiossa ja ne ovat perinteisesti olleet suhteellisen uusia ominaisuuksia. Kartoittamalla näiden tukea voitiin selvittää onko tuki ehtinyt levitä kaikille alustoille ja mahdollisesti minkä ohjelmistoversion myötä.

Omassa kategoriassaan testattiin web-kirjasinformaatit EOT[119], TTF[120]/ OTF[121], SVG ja WOFF[122], video-formaatit H.264/MPEG-4 AVC, Ogg/

[119] `http://w3.org/Submission/EOT/`

[120] `http://developer.apple.com/fonts/TTRefMan/`

[121] `http://microsoft.com/typography/otspec/`

[122] `http://w3.org/TR/WOFF/`

Theora ja WebM/BP8, sekä audio-formaatit M4A, MP3, Ogg ja WAV. Eri formaattien välisissä taisteluissa on perinteisesti nähty merkittäviä selainkenttää jakavia liittoumia. Formaatteja kartoittamalla haluttiin selvittää onko esimerkiksi H.264:n voimakas yleistyminen verkossa[1][2][3] vaikuttanut eri alustojen selainten tukeen tai onko W3C:n standardiksi ajama WOFF saanut jo pysyvän jalansijan web-kirjasinformaattien joukossa.

Alusta- ja selainkartoituksella haluttiin myös selvittää selainten tukea CSS-mediakyselyille ja kehittäjien tarvetta tukeutua selainvalmistajakohtaisiin etuliitteisiin mediakyselyitä laadittaessa. Lisäksi kiinnostavia olivat vektorigrafiikkaformaatti SVG:n tuet eri yhteyksissä, jotka helpottavat omalta osaltaan käyttöliittymien ja web-grafiikan suunnittelua eri kokoisille laitteille ja näytöille. Lisäksi kartoitettiin tuki muun muassa DOM-kosketustapahtumille, sillä aikaisemmin Stage Framework -ohjelmistokehyksen web-applikaation kehittämisen aikana oli käynyt ilmi Windows Phone -mobiilikäyttöjärjestelmän version 7.5 puutteellinen tuki kosketuksille.

6.1.2 Kartoituksen tulokset

Alusta- ja selainkartoituksen tulokset on julkaistu kokonaisuudessaan diplomityöstä erillisenä dokumenttina [15]. Kartoitetut laitteet ja selaimet on esitetty tämän diplomityön liitteessä A. Seuraavassa on käsitelty kartoituksen keskeisimmät tulokset. Kaikki testatuista laitteista tukivat HTML5-sovellusvälimuistia ja HTML5-tallennustilaa Lumia 900 -laitetta lukuunottamatta, joka tuki ainoastaan jälkimmäistä. Kaikissa Android-laitteissa ja useissa työpöytäselaimissa sovellusvälimuistin koko oli teoriassa rajaton, käytännössä raja on kuitenkin erityisesti puhelinten tapauksessa joitain satoja megatavuja. Kartoituksessa sovellusvälimuistin koko merkittiin rajattomaksi, mikäli sovellusvälimuistiin pystyttiin tallentamaan tietoa yli sadan megatavun edestä.

iOS-alustalla sovellusvälimuisti oli aikaisemmilla iOS 5 -versioilla oletuksena 5 megatavua, joka ylitettäessä käyttäjälle esitettiin kysymys välimuistin nostamisesta 10:n ja 25:n megatavun kautta aina 50 megatavuun. iOS 6 -versioiden myötä oletusvälimuistin koko nousi 25 megatavuun ja käyttäjälle esitetään kysymys ainoastaan nostosta 50 megatavuun. Testatuista työpöytäselaimista Firefox tarjosi käyttäjälle mahdollisuuden valita välimuistin suuruuden 50 megatavusta aina 1024 megatavuun ja Opera viidestä megatavusta rajoittamattomaan. Uudessa Internet Explorer 10 -selaimessa sovellusvälimuistin oletuskoko oli 10 megatavua, jonka jälkeen

[1][2][3] `http://gigaom.com/2011/12/19/h264-80-percent-of-videos/`

selain pyysi käyttäjältä välimuistin nostoa 50 megatavuun. Kaiken kaikkiaan sovellusvälimuistin oletustila ja laajennusmahdollisuus vaihteli suuresti eri selainten välillä. Käyttäjän suostumuksella sovellusvälimuistia voidaan kuitenkin tällä hetkellä käyttää alustasta ja selaimesta riippumatta 50 megatavuun saakka.

HTML5-tallennustilan suuruus oli sovellusvälimuistia konsistentimpi ja paikallisen tilan suuruus oli pääasiassa 5 megatavua muutamia poikkeuksia lukuunottamatta. Esimerkiksi työpöytäselaimista Opera tarjosi ainoastaan 3,75 megatavun paikallisen tallennustilan, joka on ehtinyt muuttua kartoituksen jälkeen julkaistuissa Opera-versioissa viideksi megatavuksi. Internet Explorer 10 -selaimen tapauksessa tilaa luvattiin paikalliselle ja sessiokohtaiselle tallennustilalle molemmille 10 megatavua. Työpöytäselaimessa tila jäi testattaessa yhteen megatavuun ja mobiiliselaimessa 9,5 megatavuun. Firefox tarjosi ainoana selaimena rajattoman paikallisen tallennustilan. Sessiokohtainen tallennustila oli Android-laitteilla rajaton ja iOS:llä 5 megatavua. Kartoituksesta päätellen alustariippumattoman HTML5-tallennustilan käyttämisessä voidaan turvautua toistaiseksi ainoastaan yhden megatavun tallennustilaan.

Kaikki testatut laitteet ja selaimet tukivat HTML5-mediaelementtejä ja HTML5-piirtoaluetta. Videoformaateista laajin tuki oli H.264/MPEG-4 AVC:llä ja audioformaateista MP3:lla. Ainoastaan työpöytäselainten Firefox ja Opera eivät tukeneet näitä formaatteja oletuksena. Toiseksi laajin tuki videoformaateista oli WebM:llä ja VP8:lla, joita tuettiin oletuksena Android-laitteiden Android Browser -selaimessa. Monet laitteet pystyvät kuitenkin hyödyntämään rautapohjaista H.264:n purkua, mikä tekee siitä ihanteellisimman formaatin HTML5-videoelementille. Ogg/Theoraa tuki testatuista mobiilikäyttöjärjestelmien selaimista ainoastaan Firefox.

CSS3-animaatiot, -transformaatiot ja -siirtymät olivat erinomaisesti tuettuja, ainoana poikkeuksena oli jälleen Windows Phone 7.5:n Internet Explorer -selain. CSS3-transformaation 3D-transformaatio ei ollut käytettävissä Opera-selaimissa, Windowsilla toimivassa Maxthon-selaimessa ja vanhemmissa Android Browser -selaimissa, Gingerbread-käyttöjärjestelmäversiosta alaspäin. Näilläkin selaimilla transformaatioita voitiin toteuttaa 2D-transformaatiolla. Kartoituksessa testattujen laitteiden ja selainten erinomainen CSS3-tuki oli lupaavaa toteutettua web-applikaatiota silmälläpitäen. Transformaatioiden ja siirtymien hyödynnettävyyttä testattiin kuitenkin vielä tarkemmin suorituskykymittauksissa.

Myös CSS-mediakyselyiden tuki oli erinomainen. Selaimet tukivat myös spesifikaatioon kuulumatonta pikselitiheys-kyselyä, jonka yhteydessä kaikki selaimet käyttivät pääasiassa -webkit-etuliitettä: ainoastaan Opera käytti

omaa -o-etuliitettä. Kartoituksessa testattiin myös perinteisesti uutena pidettyjä CSS-tyyliasetuksia, kuten varjostuksia, kulmien pyöristystä, taustakuvan koon lisäasetuksia ja useamman taustakuvan tukea. Testatut selaimet tukivat näitä kaikkia, poikkeuksena Windows Phone 7.5:n Internet Explorerin puuttuva tekstivarjostus-tuki.

Myös taustasuorittajat ja web-soketit olivat hyvin tuettuja: lähes 75% testatuista yhdistelmistä tuki molempia. Puutteellisinta tuki oli Androidilla, jolla testatuista laitteista ainoastaan Xperia T tuki molempia. Erikoista oli taustasuorittajatuen puuttuminen Galaxy S III -laitteesta, jossa oli sama ohjelmistoversio Xperia T:n kanssa. Myös WebGL-tuki oli yllättävän laajaa, ylittäen jopa CSS-tavutuksen tuen, joka puolestaan oli heikosti tuettu. Kartoituksen perusteella CSS-tavutuksen kanssa olisi tällä hetkellä käytettävä vielä varmuuden vuoksi JavaScript-kirjastoon pohjautuvaa tavutusta, mikäli sisältöä on välttämätöntä tavuttaa. WebGL-tuki puuttui harmillisesti uudesta Internet Explorer 10 -selaimesta niin työpöytä- kuin mobiilipuolellakin.

Vektorigrafiikkaformaatti SVG:n tuki oli erittäin hyvä. Ainoastaan Opera ei tukenut SVG:tä *img*-elementissä käytettäessä. Androidilla SVG-tuki puuttuu Gingerbread-versiosta ja sitä vanhemmista ohjelmistoversioista. SVG:tä voi kuitenkin turvallisesti käyttää modernien selainpohjaisten sovellusten osana, jotka tukeutuvat muutenkin esimerkiksi CSS3:n eri tekniikoihin. Kartoituksen viimeisenä kuriositeettina mainittakoon testatun Windows Phone 8 -laitteen puuttuva kosketustapahtumatuki. Laitteen Internet Explorer 10 -selain tukee kosketuksia paremmin kuin aikaisempien Windows Phone -mobiilikäyttöjärjestelmien selaimet, mutta DOM-kosketustapahtumien toteutus ei vastaa spesifikaatiota. Puutteellinen kosketustapahtumatuki vaikeuttaa alustariippumattomien web-applikaatioiden toteuttamista.

Web-kirjasimista tuetuimmat kirjasintyypit olivat TTF ja OTF sekä näiden jälkeen WOFF. Kirjasintyyppien ja eri mediaformaattien tuki on esitetty tämän diplomityön liitteessä B.

6.1.3 CSS3-kirjasinmoduulin kartoitus

Yksi CSS3:n uusista ominaisuuksista on laajennettu kirjasintuki, joka on tuonut mukanaan lukuisia uusia asetuksia. Tuki näille asetuksille on useimmissa selaimissa vielä alkutekijöissään, mutta web-kirjasinten laaja tuki nykyselaimissa antaa toveita nopeasta yleistymisestä. CSS3-kirjasinmoodulin selaintuki kartoitettiin eri asetuksien osalta erikseen iOS, Android ja Windows Phone -alustojen oletusselainten uusimmille versioille käyttäen The

CSS3 Test -palvelua[124]. Palvelu ei tarkista tuettuja asetuksia konkreettisesti, vaan tarkastelee yksinomaan paljastaako selaimen rajapinta kyseistä asetusta, eli hyväksyykö selain asetuksen[124]. Selainvalmistajat eivät kuitenkaan yleensä lisää asetuksia käytettäväksi ellei niiden taustalta löydy valmista toteutusta. Muussa tapauksessa selainvalmistaja joutuu nopeasti huonoon valoon kehittäjien keskuudessa.

Vaikka kirjasinominaisuuksien tuki on vielä keskeneräinen, löytyy eri alustoilta sekalaista tukea ja ensimmäisiä implementaatioita. Kirjasinominaisuusasetuksia (engl. font feature properties) on toteutettu tässä vaiheessa ahkerimmin iOS-alustan Safari-selaimelle, joka tukee ainoana alustana kokonaan kirjasinvälistys-asetusta, sekä osittain ligatuureja. Seuraavissa kuvissa 6.2-6.5 on esitetty iOS:n Safarin kolme tukemaa ja yksi tukematon ligatuuri-asetus antaen esimakua web-kirjasinten tarjoamista mahdollisuuksista. Taulukko kaikkien kirjasinasetusten tuen tilasta on esitetty erikseen julkaistussa tulosdokumentissa [15]. Taulukko sisältää myös muita kirjasinten ulkonäköön vaikuttavia CSS-ominaisuuksia.

Kuva 6.2: Yleiset ligatuurit (engl. common ligatures). [23]

Kuva 6.3: Valinnaiset ligatuurit (engl. discretionary ligatures). [23]

Kuva 6.4: Historialliset ligatuurit (engl. historical ligatures). [23]

[124] `http://css3test.com/`

Kuva 6.5: Kontekstuaaliset ligatuurit (engl. contextual ligatures). [23]

Kuvassa 6.2 esitetään yleiset ligatuurit, jotka ovat käytössä automaattisesti kaikilla OpenType-kirjasimilla tuen implementoinnin jälkeen. Ligatuurit korvaavat kaksi toistensa kanssa sekoittuvaa merkkiä yhdellä kirjasimessa määritellyllä hallitulla merkillä. Kuvassa 6.3 on esimerkki valinnaisista ligatuureista, jotka ovat yleensä kirjasimen suunnittelijan koristeellisia lisäyksiä kirjasimen ilmeeseen. Kuvassa 6.4 esitetyt historialliset ligatuurit esittävät kirjasinkäytäntöjä, jotka ovat olleet tyypillisiä historian saatossa, mutta eivät välttämättä ole luettavuudeltaan paras vaihtoehto tekstille. Nämä kolme ligatuuriasetusta ovat käytettävissä iOS-alustoilla, mutta eivät toistaiseksi muualla. Kuvassa 6.5 esitetyt kontekstuaaliset ligatuurit, jotka valitsevat käytetyt merkit ympäröivien merkkien mukaan on toistaiseksi ainoa, jolle ei löydy tukea myöskään iOS:llä.

CSS3-kirjasinmoduulin kartoituksen tulokset on esitetty kokonaisuudessaan alusta- ja selainkartoituksen tuloksista julkaistun dokumentin osana [15].

6.2 Suorituskykymittaukset

Alusta- ja selainkartoituksen lisäksi diplomityön osana suoritettiin suorituskykymittaukset useille laitteille eri alusta- ja selainyhdistelmillä. Suorituskykymittausten tarkoituksena oli osaltaan kartoittaa laitteiden valmiutta uusien HTML5- ja CSS3-teknologioiden käyttöön, mutta myös vertailla eri laitteistoja, ohjelmistoja ja ohjelmistoversioita. Suorituskykymittaukset suoritettiin 46:lle alusta- ja selainyhdistelmälle. Selaimien suorituskykyä mittaavia ohjelmistokirjastoja on olemassa useita, muun muassa Googlen Octane[125] ja sitä edeltänyt V8 Benchmark Suite[126], sekä Mozillan Kraken[127] ja Dromaeo[128], mutta tämän suorituskykymittauksen tarkoituksena oli keskittyä kehitetyn web-applikaation ongelmakenttään. Testejä toteutettaessa on sovellettu käytäntöjä muista ohjelmistokirjastoista, mutta arkkitehtuuriltaan testit ovat perustuneet yleisesti käytettyyn toiminnon suorit-

[125] http://developers.google.com/octane/
[126] http://v8.googlecode.com/svn/data/benchmarks/v7/run.html
[127] http://krakenbenchmark.mozilla.org/
[128] http://dromaeo.com/

tamiseen kuluvan ajan mittaamiseen - vaihtoehtona olisi ollut myös esimerkiksi suorituskertojen mittaaminen valitussa ajassa [64].

6.2.1 Suoritetut testit

Erilaisia testejä suoritettiin yhteensä kymmenen, joista jokainen testi suoritettiin 60 kertaa. Suorituskykymittausten keskiarvoja laskettaessa pudotettiin tuloksista pois viisi nopeinta ja viisi hitainta arvoa, jolloin keskiarvo laskettiin 50 mittausarvon perusteella. Alunperin testejä suoritettiin 120 kappaletta ja merkitseviä arvoja mitattiin 100, mutta määrää jouduttiin pienentämään testien raskauden vuoksi erityisesti matkapuhelimia varten, joilla esiintyi satunnaista selainten epävakautta. Vaikka testien määrää jouduttiin rajoittamaan käytännöllisistä syistä, on valittu määrä riittävä tarkasteltaessa hajontojen ja luottamuvälien kehittymistä testien määrää laskettaessa. Nopeimpien ja hitaimpien arvojen poistamisella haluttiin omalta osaltaan parantaa lopullisten tulosten laatua rajoittamalla satunnaisten ääriarvojen vaikutusta. Testit koostuivat pääasiassa HTML5-teknologioiden suorituskykyä mittaavista testeistä, mutta myös kahdesta CSS3-testistä, jotka ovat keskeisiä diplomityössä toteutetun Stage Framework -ohjelmistokehyksen toiminnalle. Seuraavaksi on kuvattu jokainen suorituskykymittauksen testi. Kuvauksissa viitatut testikuvat on esitelty kuvausten jälkeen kuvassa 6.6, sekä tarkemmin taulukossa 6.7.

6.2.1.1 HTML5-sovellusvälimuistitesti

HTML5-sovellusvälimuistin osalta (engl. HTML5 Application Cache) testattiin yhden tavallisen (ei-Retina ja base64-enkoodaamattoman) kuvan hakunopeutta palvelimelta välimuistiin. Kuva oli Aalto University Magazinen neljännen numeron kansikuva PNG-formaatissa, joka oli noin yhden megatavun suuruinen. Testissä käytettiin diplomityössä kehitettyä dynaamista HTML5-sovellusvälimuistia, jossa välimuistitettavat resurssit asiakaskohtaisesti listaava manifest-tiedosto päivitetään käyttöliittymän pyynnöstä. Testissä suoritettiin POST-pyyntö palvelimelle, jossa välitettiin aina uusi välimuistitettava resurssi. Resurssina toimi todellisuudessa ainoastaan yksi PNG-kuva, mutta selaimet pakotettiin lataamaan se joka kerta uutena resurssina vaihtamalla resurssin nimen perässä esiintyvää GET-parametria. AJAXilla tehdyn resurssin päivityksen jälkeen selaimelle annettiin käsky päivittää HTML5-sovellusvälimuisti. Välimuistin onnistuneen päivityksen jälkeen yksi testisuoritus päättyi.

6.2.1.2 HTML5-tallennustilatesti 1

HTML5-tallennustilan (engl. Web Storage) testaaminen oli lähtökohtaisesti hankalampaa kuin HTML5-sovellusvälimuistin, sillä selainten tukema tallennustila on suhteellisen pieni. Toisena ongelmana oli HTML5-tallennustilan tuki ainoastaan tekstimuotoiselle datalle. Tallennustila toimii hyvin nopeasti, joten pienten tekstimäärien tallentamisen testaamisessa ei ollut mieltä jo todella suureksi kasvavan virheen osuuden vuoksi. HTML5-tallennustilalle päätettiin loppujen lopuksi suorittaa kaksi testiä: ensimmäisessä testataan valmiiksi base64-enkoodatun kuvan tallentamista tallennustilaan noutamalla se ensin palvelimelta, kun taas toisessa palvelimelta haetaan alkuperäinen kuva, joka base64-enkoodataan selaimessa HTML5-piirtoalueella (engl. HTML5 Canvas Element) ja tallennetaan tämän jälkeen tallennustilaan. Näin toimimalla saatiin kaksi eri tavalla vertailukelpoista tulosta HTML5-sovellusvälimuistitestille.

6.2.1.3 HTML5-tallennustilatesti 2

Toisessa HTML5-tallennustilatestissa palvelimelta noudettiin tavallinen kuva (ei-Retina ja base64-enkoodaamaton), joka muutettiin base64-muotoon HTML5-piirtoalueen *data URL* -ominaisuudella. Tämän jälkeen kuva tallennettiin tekstimuodossa tallennustilaan. Kuten taulukosta 6.7 huomataan, HTML5-tallennustilatestit eroavat myös toisella tapaa toisistaan: ensimmäisessä testissä siirrettävän datan määrä on 1,6 megatavua base64-muodon suuremmasta tilantarpeesta johtuen verrattuna toisen testin yhteen megatavuun, joka vasta selaimessa muuttui 1,6 megatavuksi. Lopullinen tallennustilalta vaadittu tilantarve oli siis molemmissa testeissä sama.

6.2.1.4 Taustasuorittajatesti 1

Taustasuorittajien (engl. Web Workers) ensimmäisessä testissä testattiin kolmen tavallisen kuvan muuntamista base64-muotoon peräkkäin suoritettavissa taustasuorittajissa. Testi toimi vertailutestinä kahteen muuhun taustasuorittajatestiin, joista ensimmäisessä muuttuvana tekijänä toimi kuvan suuruus ja toisessa suoritustapa.

6.2.1.5 Taustasuorittajatesti 2

Taustasuorittajien toisessa testissä testattiin yhden Retina-laatuisen kuvan muuntamista base64-muotoon käyttäen yhtä taustasuorittajaa. Testillä ha-

luttiin selvittää muuntoon kuluvaan aikaan vaikuttavia tekijöitä, sillä yksi Retina-kuva vastaa pikselimäärältään neljää ja kooltaan noin kolmea tavallista kuvaa. Ensimmäisessä testissä muunnettiin kolme tavallista kuvaa peräkkäin.

6.2.1.6 Taustasuorittajatesti 3

Taustasuorittajien kolmannessa ja viimeisessä testissä testattiin kolmen tavallisen kuvan muuntamista base64-muotoon käyttäen kolmea taustasuorittajaa yhtä aikaa. Testillä haluttiin selvittää hyötyykö taustasuorittajien käytöstä verrattain pienellä laskennallisella operaatiolla. Hyvin monissa mobiililaitteissa on jo käytössä moniydinsuorittimia, mikä ei automaattisesti tarkoita suorituskykyetua taustasuorittajien kohdalla, mutta voi vaikuttaa suorituskykyyn myönteisesti.

6.2.1.7 HTML5-piirtoaluetesti 1

HTML5-piirtoalueen (engl. HTML5 Canvas Element) ensimmäisessä testissä testattiin tavallisen kuvan muuntamista base64-muotoon käyttäen piirtoalueen *data URL* -ominaisuutta. Toimenpide on sama kuin HTML5-tallennustilan toisessa testissä, mutta tällä kertaa ilman verkon ja tallennustilan vaikutusta. Käytännössä testissä ladattiin PNG-muotoinen Aalto University Magazinen kansikuva piirtoalueeseen, jonka jälkeen piirtoalueelta pyydettiin kuvan *data URL* -muotoa, joka vastaa muuntamista ja esittämistä base64-enkoodatussa muodossa.

6.2.1.8 HTML5-piirtoaluetesti 2

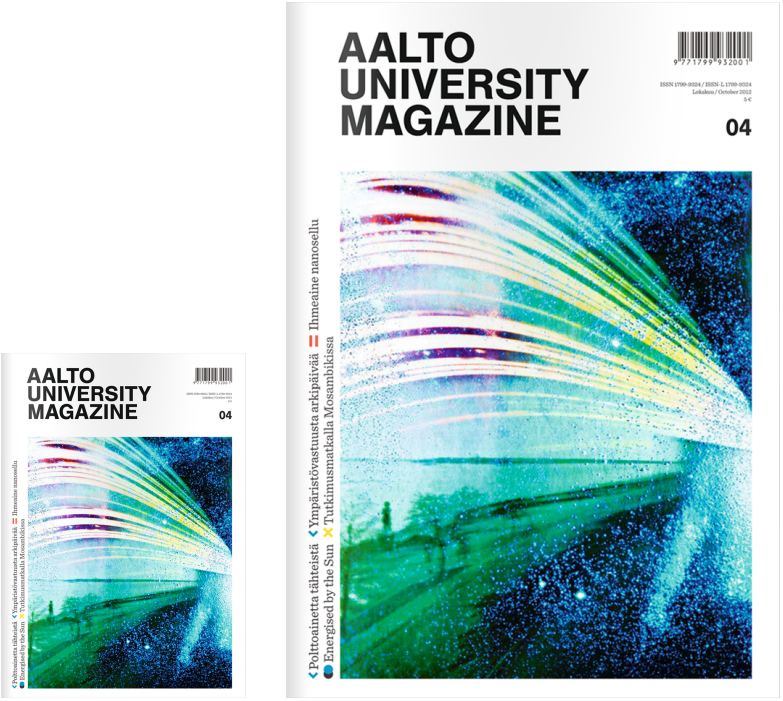
HTML5-piirtoalueen toisessa testissä kiinteän kokoiselle piirtoalueelle renderöitiin ympyrägradientti, jonka päälle generoitiin 100 tekstielementtiä mustalla varjostuksella ja tekstin täyttävällä ympyrägradientilla. Tekstielementeille annettiin satunnainen koko ja sijainti piirtoalueella, sekä käännettiin kiinteään astemäärän verran iteraatioiden välillä. Verrattuna ensimmäiseen piirtoaluetestiin, jälkimmäinen testi testaa piirtoaluetta autenttisemmin sen alkuperäistä käyttötarkoitusta ajatellen.

6.2.1.9 CSS3-transformaatiotesti

CSS3-transformaatiotesti mahdollistettiin sekä 2D- että 3D-transformaatioille. Selaimet, jotka eivät tukeneet 3D-transformaatiota erotettiin tuloksissa muista selaimista. CSS3-transformaatiotestissä generoitiin 1000 tekstielementtiä vaihtelevalla läpinkäyvyydellä, sijainnilla ja käännöllä. Yhden transformaatiotestin suorituksen aikana näiden generoitujen tekstielementtien ominaisuuksia muutettiin aiheuttaen 1000 elementin päivittymisen CSS3-transformaatiolla. Testi toimii hyvin samaan tapaan kuin miten CSS3-transformaatiota käytetään Stage Framework -ohjelmistokehyksen selausnäkymässä ja monissa referenssijulkaisuissa sivujen liikuttamiseen kun käyttäjä koskettaa sivua. Testillä haluttiin keskittyä erityisesti suuren elementtimäärän aiheuttamaan suorituskykyvaikutukseen.

6.2.1.10 CSS3-siirtymätesti

CSS3-siirtymätestissä testattiin suorituskykyä tilanteessa, jossa selain animoi itsenäisesti elementtien sijaintia perustuen CSS3-siirtymiin (engl. CSS3 Transitions). Stage Framework -ohjelmistokehyksen ja eräiden referenssijulkaisujen tapauksessa tämä tarkoittaa tilannetta, jossa käyttäjä pyyhkäisee sivua siirtyäkseen seuraavalle ja päästää irti kosketuksen - selain animoi sivun oikealle kohdalleen. Siirtymätestissä käytetiin samoja 1000 tekstielementtiä kuin transformaatiotestissä, mutta tällä kertaa ainoastaan elementtien läpinäkyvyyttä animoitiin siirtymien avulla. Testillä haluttiin osittain selvittää DOM-rakenteen päivityksestä aiheutuvaa suorituskykyvaikutusta, mikä vaikuttaa myös CSS3-transformaatiotestissä. Kuten myös transformaatiotesti ja osa muista testeistä, myös siirtymätesti olisi voitu suorittaa käänteisesti hakemalla kiinteällä maksimisuoritusajalla suurinta elementtimäärää, jota voidaan manipuloida kasvattaen vähitellen testattua elementtimäärää. Johtuen virheen suhteesta nimenomaan testin suoritusaikaan nähtiin suorituskykymittauksissa paremmaksi käyttää kiinteää elementtimäärää, jonka mainpulointiaikaa verrataan - näin myös testeistä saatiin riittävän pitkiä jokaiselle testiyksikölle.



Kuva 6.6: Suorituskykytesteissä käytetyt kuvat Aalto University Magazinen neljännen numeron kannesta[129]. Kuvien kokoero ilmentää tavalliselle ja pikselitiheydeltään kaksinkertaiselle laitteelle tarvittavia kuvatarkkuuksia.

Taulukko 6.7: Suorituskykytesteissä käytettyjen kuvien tiedot.

Tunniste	Tarkkuus	Pikselimäärä	Koko (PNG)	Koko (base64)
cover	681x962	655 122	1 Mt	1,6 Mt
cover@2x	1362x1925	2 621 850	3 Mt	5,1 Mt

6.2.2 Testijärjestely

Suorituskykymittaukset oli alunperin suunniteltu järjestettäväksi Aalto-yliopiston avoimessa Aalto open -langattomassa verkossa käyttäen mediatekniikan laitoksen palvelinta. Tässä testijärjestelyssä tiedonsiirtoa vaati-

[129] `http://aalto.fi/fi/current/magazine/04/`

vat testit olisivat laittaneet testattavat laitteet ja alustat luonnollisempaan käyttöympäristöön digitaalisten julkaisujen käyttämisen kannalta. Hyvin nopeasti kävi kuitenkin selväksi, että vapaassa käytössä olevan langattoman verkon ja mediatekniikan laitoksen web-palvelimen vaihteleva kuorma vaikutti testituloksiin liian voimakkaasti. Tämä voitiin todeta testaamalla sama laite useaan kertaan eri ajankohtina yhden vuorokauden sisällä. Käyttämällä etäällä olevaa tukiasemaa myös laitteiden WLAN-sovittimien suorituskykyeroista näytti tulevan liian hallitseva tekijä tutkittaessa eroja eri ohjelmistoteknologioiden välillä. Näiden syiden valossa suorituskykymittausten testijärjestelyä muutettiin orjallisemmaksi.

Suorituskykymittauksissa käytettiin kahta testijärjestelyä, ensisijaisen laiteryhmän muodostivat kannettavat laitteet eli matkapuhelimet, tabletit ja kannettavat tietokoneet, kun taas toiseen laiteryhmään kuuluivat kaksi testattua pöytäkonetta. Testitulosten vertailun mielekkyyden kannalta pöytäkoneiden laiteryhmän testijärjestelystä pyrittiin kuitenkin saamaan mahdollisimman vertailukelpoinen kannettavien laitteiden kanssa. Muun muassa testatun kannettavan tietokoneen selainten suorituskyky on lähempänä pöytäkoneiden selainten suorituskykyä kuin muiden kannettavien laitteiden.

Molemmissa testijärjestelyissä käytettiin palvelinkoneena yhtä Applen 2011-vuoden lopulla julkaistua MacBook Pro -tietokonetta, joka oli varustettu kaksiytimisellä 2.4GHz Intel Core i5 -suorittimella, 4GB DDR3-muistilla ja 5400 kierrosta minuutissa pyörivällä HDD-kiintolevyllä. Käyttöjärjestelmänä oli 64-bittinen Apple OS X 10.8.2 Mountain Lion ja web-palvelinsovelluksena Apache 2.2.22. Kannettavien laitteiden testijärjestelyssä palvelinkone oli kytketty verkkovirtaan ja testattavat laitteet toimivat akulla, kun taas pöytäkoneiden testissä palvelinkone toimi akkuvirralla ja pöytäkoneet oli kytketty verkkovirtaan. Molemmissa testijärjestelyissä palvelinkone jakoi Ethernet-kaapelin kautta tulevan Internet-yhteyden OS X:n verkkojaolla 5GHz:n langattomalle lähiverkolle WPA2-salauksella. Testijärjestelyn kannalta on olennaista mainita Internet-yhteyden jakaminen, vaikka testilaitteet eivät sitä tarvinneet, sillä se vaikuttaa OS X:n verkkojaon toimintaan.

Suorituskykymittaukset suoritettiin käyttäen neljännesmetrin etäisyyttä palvelinkoneen ja testattavien laitteiden välillä. Testattavat laitteet yhdistivät palvelinkoneen luomaan langattomaan verkkoon ja käyttivät tämän yhteyden kautta palvelinkoneen web-palvelimella sijainneita resursseja. Pöytäkoneiden testijärjestelyssä pöytäkoneet oli kytketty Ethernet-kaapelilla Apple AirPort Extreme -tukiasemaan, joka toimi langattoman lähiverkon yhteyspisteenä. Tässä testijärjestelyssä palvelinkone yhdisti tukiaseman luomaan 5GHz:n WPA2-salattuun langattomaan lähiverkkoon samalta neljän-

nesmetrin etäisyydeltä. Tällä järjestelyllä langattoman lähiverkon vaikutus voitiin vakioida siedettävälle tasolle.

6.2.3 Testitulokset

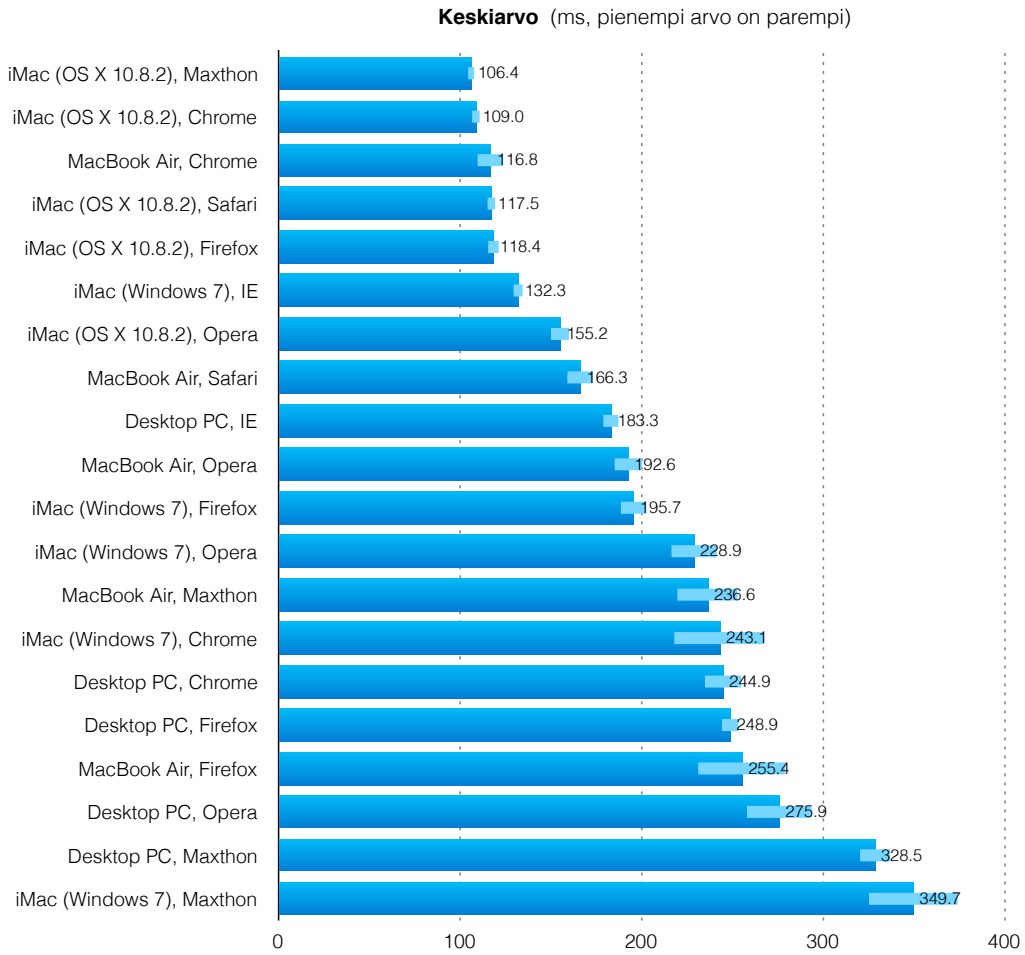
Suorituskykymittauksista saatiin suuri määrä aineistoa - jokaista testiä suoritettiin 46:lle alusta- ja selainkombinaatiolle 60 kertaa, jolloin testituloksia saatiin kaikkiaan 27 600 kappaletta: todellinen luku on kuitenkin hieman vähemmän johtuen muun muassa joitakin teknologioita tukemattomista selaimista. Kaikkia tuloksia ei voida tässä diplomityössä esittää, mutta tähän lukuun on poimittu tutkimusaiheen kannalta mielenkiintoisimpia kokonaisuuksia. Testitulokset on julkaistu kokonaisuudessaan erillisenä tulosdokumenttina [60]. Testatuista mobiilikäyttöjärjestelmistä suurinta osaa edustivat iOS ja Android -laitteet, mutta testeissä oli myös mukana Windows Phone -laitteita. Windows Phone -laitteista ainoastaan Windows Phone 8 -ohjelmistoversiolla varustettu laite tuki kaikkien testien vaatimia teknologioita, joten Windows Phone 7.5 -ohjelmistoversiolla varustettu Nokia Lumia 900 löytyy ainoastaan HTML5-tallennustila- ja -piirtoaluetesteistä. Testatut laitteet on listattu diplomityön liitteessä A.

Testitulosten virheen osuuden pienentämiseksi jokainen testeistä oli suunniteltu kestämään tabletti- ja mobiililaitteilla vähintään noin 100 millisekuntia. Erityisesti vanhoilla selaimilla ja käyttöjärjestelmillä selaimen antama aika, jolla testien suoritusta tarkkaillaan voi heittää jopa 15 millisekuntia, kuitenkin keskimäärin noin 7,5 millisekuntia[130]. Tilanne on kuitenkin parantunut huomattavasti alustojen ja selaimien kehittyessä, ja vastaavansuuruisia aikaan liittyviä virheitä ei havaittu testatuilla alustoilla ja selaimilla. W3C:n Web Performance Working Group valmistelee User Timing -spesifikaatiota, joka antaisi kehittäjille pääsyn korkean tarkkuuden aikaleimoihin uusilla *PerformanceMark* ja *PerformanceMeasure* -rajapinnoilla, jotka laajentavat *Performance*-rajapintaa [65]. Selainten implementoidessa spesifikaatiossa kuvatut rajapinnat, muuttuisi ajanotosta aiheutuva virhe häviävän pieneksi, mikä puolestaan mahdollistaisi myös nopeampien testien toteuttamisen.

Kuvassa 6.8 ja 6.9 on esitetty HTML5-sovellusvälimuistitestin tulokset ensin työpöytäselaimille ja sitten tabletti- ja mobiiliselaimille. Sovellusvälimuistin tapauksessa parhaissa asetelmissa testeissä olivat OS X -käyttöjärjestelmällä toimivat WebKit-pohjaiset selaimet. Myös työpöytälaitteella toimiva Internet Explorer 10 menestyi hyvin. HTML5-selaimen maineessa olevalla Maxthon-selaimella esiintyi erityistä hitautta sovellusvälimuistin käytössä Windows-

[130] `http://ejohn.org/blog/accuracy-of-javascript-time/`

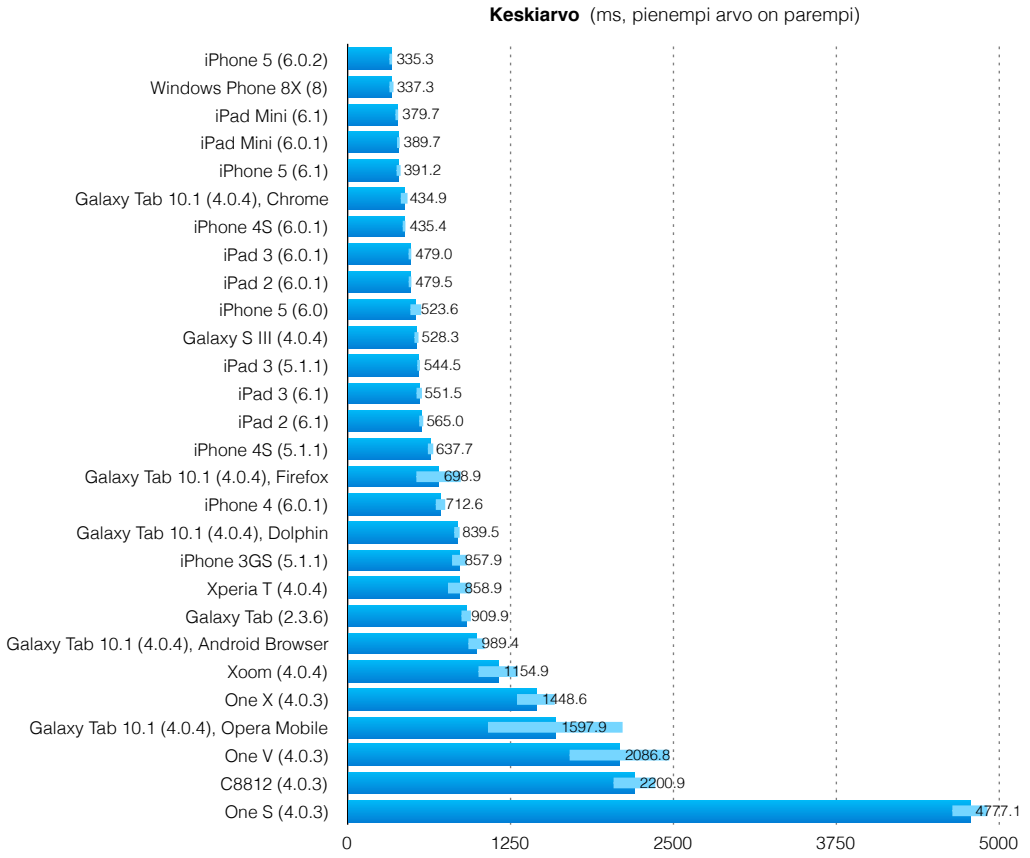
alustoilla.



Kuva 6.8: Työpöytäselaimet HTML5-sovellusvälimuistitestissä. Keskimääräinen testin suoritusaika, 95%-luottamusväli on esitetty vaaleammalla sinisellä.

Tabletti- ja mobiililaitteista parhaissa asetelmissa olivat iPhone 5 ja iPad Mini yhdessä Windows Phone 8X:n kanssa. iPhone 4S -laitteella nähtiin sovellusvälimuistin suorituskyvyn parantuvan yli 200 millisekuntia ohjelmistoversion päivittyessä versiosta 5.1.1 versioon 6.0.1. Ohjelmistoversioiden välillä nähtiin kuitenkin myös maltillista suorituskyvyn heikentymistä, esimerkiksi iPhone 5:n ohjelmistoversiosta 6.0.2 versioon 6.1. Android-alustan selaimista nopeimman sovellusvälimuistin tarjosivat Firefox ja Dolphin, joskin Firefoxin testimittaukset tuottivat verrattain suuren luottamusvälin.

Molemmat selaimet kuitenkin jättivät Android-alustan oletusselaimen Android Browserin taakseen. HTML5-sovellusvälimuistitestissä huomattavaa oli myös parhaan mobiililaitteen tulos, joka ylsi Windows-alustalla toimivien Maxthon-työpöytäselainten tulosten tasolle.

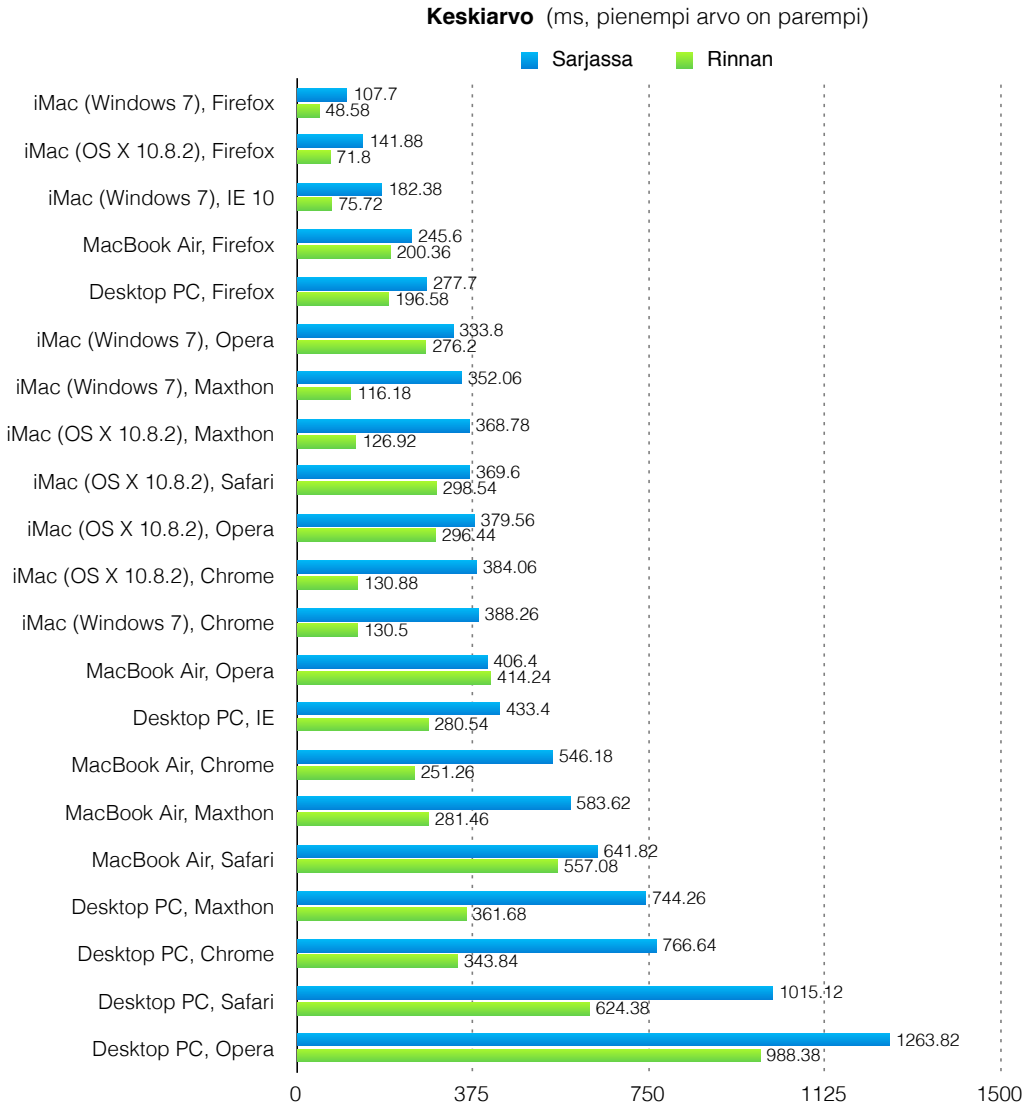


Kuva 6.9: Tabletti- ja mobiililaitteet HTML5-sovellusvälimuistitestissä. Keskimääräinen testin suoritusaika, 95%-luottamusväli on esitetty vaaleammalla sinisellä.

HTML5-tallenustilatesteissä kärkisijoja pitivät samaan tapaan OS X -käyttöjärjestelmän WebKit-pohjaiset selaimet ja Internet Explorer 10 -työpöytäselain. Myös tabletti- ja mobiililaitteissa nopeimpia tallennustilan käyttäjiä olivat iPhone 5 ja iPad Mini. Välittömästi iPad Minin perässä tuli hieman yllättäen Lumia 900 Windows Phone 7.5 -mobiilikäyttöjärjestelmällä, joka kuitenkin jäi iPad 2 ja iPad 3 -laitteiden taakse toisessa tallennustilatestissä, jossa testattiin myös kuvan reaaliaikaista enkoodausta HTML5-piirtoaluetta

käyttäen. Piirtoaluetta käyttäneen testin tuloksissa on iOS-alustan ohjelmistoversioiden välillä tapahtunut parannusta, erityisesti iPhone 5 -laitteen ohjelmistoversiosta 6.0 versioon 6.1. iPhone 5:n tapauksessa toisen tallennustilatestin tulokset näyttävät olevan konsistentimpia eri ohjelmistoversioiden välillä kuin esimerkiksi sovellusvälimuistin tapauksessa. Vanhempi Android-tabletti Motorola Xoom oli iPadien läheisyydessä ensimmäisessä tallennustilatestissä, mutta jäi piirtoalueen mukaantulon jälkeen vauhdista. Windows-alustalla toimivien Maxthon-työpöytäselainten ongelmat jatkuivat myös tallennustilatesteissä. [60]

Kuvassa 6.10 on esitetty työpöytäselainten tulokset ensimmäisessä ja kolmannessa taustasuorittajatestissä. Ensimmäisessä testissä selaimet enkoodasivat kolme noin yhden megatavun kuvaa base64-muotoon käyttäen JavaScript-toteutusta ja taustasuorittajia sarjassa. Jokainen kuva muunnettiin siis vuorollaan, kun taas kolmannessa testissä samat kuvat enkoodattiin käyttäen rinnakkaisia taustasuorittajia. Sekä työpöytäselaimista että myöhemmin kuvassa 6.11 esiteltävistä tabletti- ja mobiiliselaimista parhaiten laitteen suorittimen suorituskyvyn pystyi hyödyntämään Firefox-selain. Myös Internet Explorer 10 menestyi hyvin.



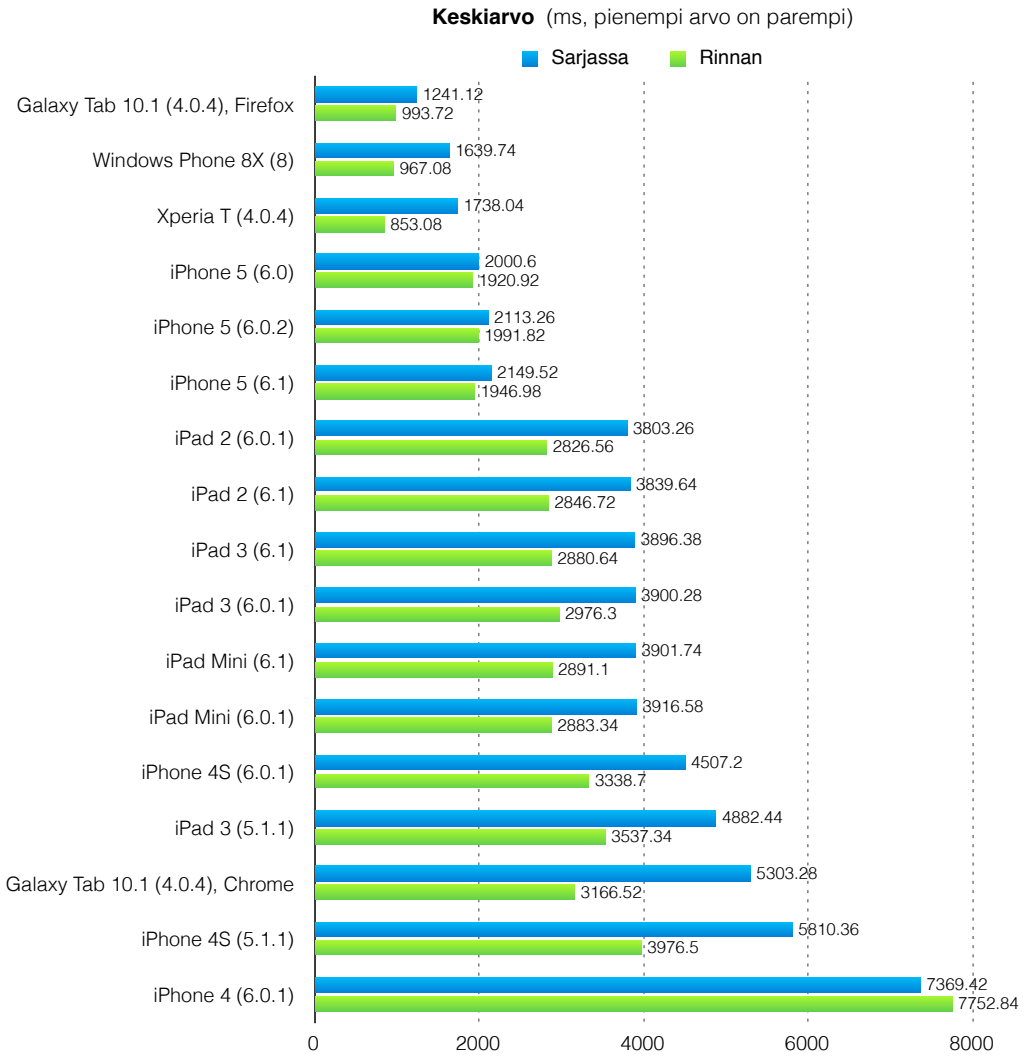
Kuva 6.10: Työpöytäselaimet taustasuorittajien ensimmäisessä ja kolmannessa testissä. Sinisellä merkityssä ensimmäisessä testissä enkoodattiin kolme tavallista kuvaa base64-muotoon sarjassa, kun taas vihreällä merkityssä kolmannessa testissä toteutettiin enkoodaus rinnan käyttämällä taustasuorittajia.

Lähes kaikki testatut laite- ja selainkombinaatiot hyötyivät rinnakkain käytetyistä taustasuorittajista. Poikkeuksina olivat yksi Opera-selaimen tulos sekä iPhone 4, joka oli testatuista laitteista ainoa yksiytimisellä suorittimella varustettu taustasuorittajia tukeva laite. iPadien testitulos parani noin

25% rinnakkaisia taustasuorittajia käytettäessä. Parhaimmillaan saatu hyöty oli neliydinsuorittimella varustetussa iMac-laittessa Chrome-selaimella, jolla testitulos parani yli 65%. Testituloksen parantuminen oli suuruudeltaan samaa luokkaa käytetystä käyttöjärjestelmästä riippumatta (OS X 10.8.2 ja Windows 7). Raa'assa laskennassa menestyivät Windows Phone 8X ja ja Xperia T, jotka nousivat iPhone 5:n ohitse. iPhone 5 ei myös juurikaan hyötynyt rinnakkaisten taustasuorittajien käytöstä, mikä pidensi Windows Phone 8X:n ja Xperia T:n etumatkaa.

Vertaillut taustasuorittajatestit olivat yllättäviä, sillä rinnakkain käytetyt taustasuorittajat paransivat suorituskykyä jo todella pienellä kuormalla, kolmea yhden megatavun kuvaa käytettäessä. Kolmen kuvan enkoodaaminen on kuitenkin käyttöliittymälle verrattain raskas operaatio kestäen nopeimmallakin mobiililaitteella reilusti yli sekunnin. Testitulokset osoittavat taustasuorittajien tekevän juuri sen mitä niiltä odotetaankin: tämän lisäksi selainten taustasuorittajaimplementaatiot näyttävät olevan teknisesti hyvätasoisia ja siten hyödynnettävissä täysimittaisesti jo tällä hetkellä.

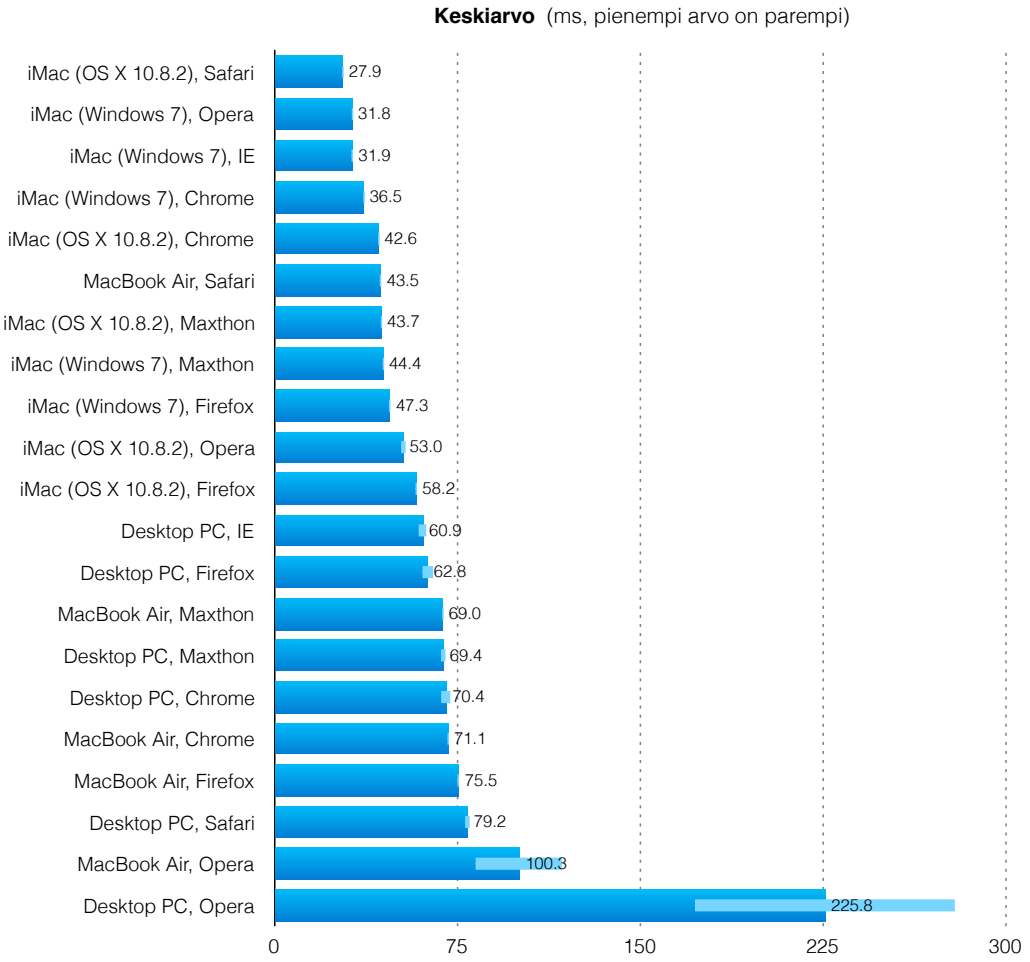
Diplomityössä toteutetun web-applikaation kannalta yksi keskeisimmistä testeistä oli CSS3-transformaatiotesti. Web-applikaation lehtien selausnäkymän toiminta tukeutuu voimakkaasti transformaatioiden käyttöön, mikä motivoi laitteiden ja selainten suorituskyvyn kartoittamiseen transformaatiota käyttäen. Transformaatiotestissä käytettiin 3D-transformaatiota, joka mahdollisuuksien mukaan hyödyntää laitteen näytönohjainta. Tämä GPU-keskeinen testi oli siis hyvää vastapainoa esitellyille CPU-painotteisille taustasuorittajatesteille. Testatuista selaimista Opera-selaimet ja Galaxy Tab (Android 2.3.6) -laitteen Android Browser -selain eivät tukeneet 3D-transformaatiota: näillä laitteilla testit suoritettiin käyttäen 2D-transformaatiota.



Kuva 6.11: Tabletti- ja mobiililaitteet taustasuorittajien ensimmäisessä ja kolmannessa testissä.

Kuvassa 6.12 ja 6.13 on esitetty CSS3-transformaatiotestin tulokset ensin työpöytäselaimille ja tämän jälkeen tabletti- ja mobiiliselaimille. Transformaatiotestiä hallitsi molemmissa kategorioissa suvereenisti Safari-selain, mikä viittaa Applen panostukseen laitteiden graafisen suorituskyvyn hyödyntämiseen selaimellaan. Graafinen suorituskyky on myös ollut avainasemassa iOS-laitteiden menestyksessä verrattuna muihin tabletti- ja mobiililaitteiden valmistajiin. Selaintoteutuksissa pelkällä JavaScriptin osuudella on suorituskyvyn kannalta enää suhteellisen pieni merkitys: yhtä tärkeää on koko ren-

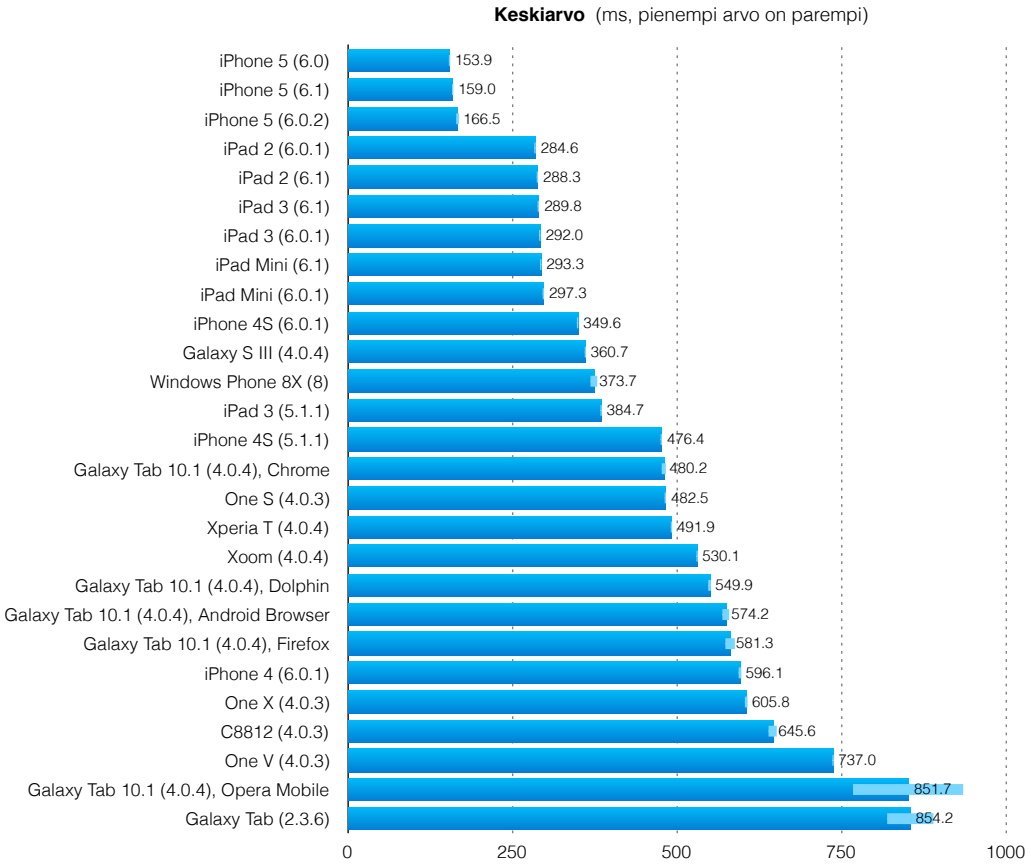
deröintipinon optimointi ja natiivien rajapintojen, kuten DOM-mallin, grafiikkamoottoreiden ja rautapohjaisen grafiikkakiihdytyksen tehokkaat implementaatiot [66].



Kuva 6.12: CSS3-transformaatiotesti, työpöytäselaimien keskimääräinen suoritusaika. 95%-luottamusvälit merkitty vaaleammalla sinisellä.

Apple on panostanut iPhone 5 -laitteensa GPU:hun, sillä laite suoriutuu CSS3-transformaatiotestistä alle puolella siitä ajasta, mitä testin suorittamiseen kuluu aikaisemman sukupolven iPhone 4S -laitteelta. Suorituskyvyn parantaminen ei kuitenkaan jää ainoastaan raudan tasolle, sillä sekä iPhone 4S että iPad 3 -laitteilla havaitaan merkittävät CSS3-transformaatiotestin suoritusajan parannukset siirryttäessä iOS 5.1.1 -ohjlemistoversiosta 6.x-versioihin. CSS3-transformaatiotestissä iOS-laitteita lähimmäksi pääsevät

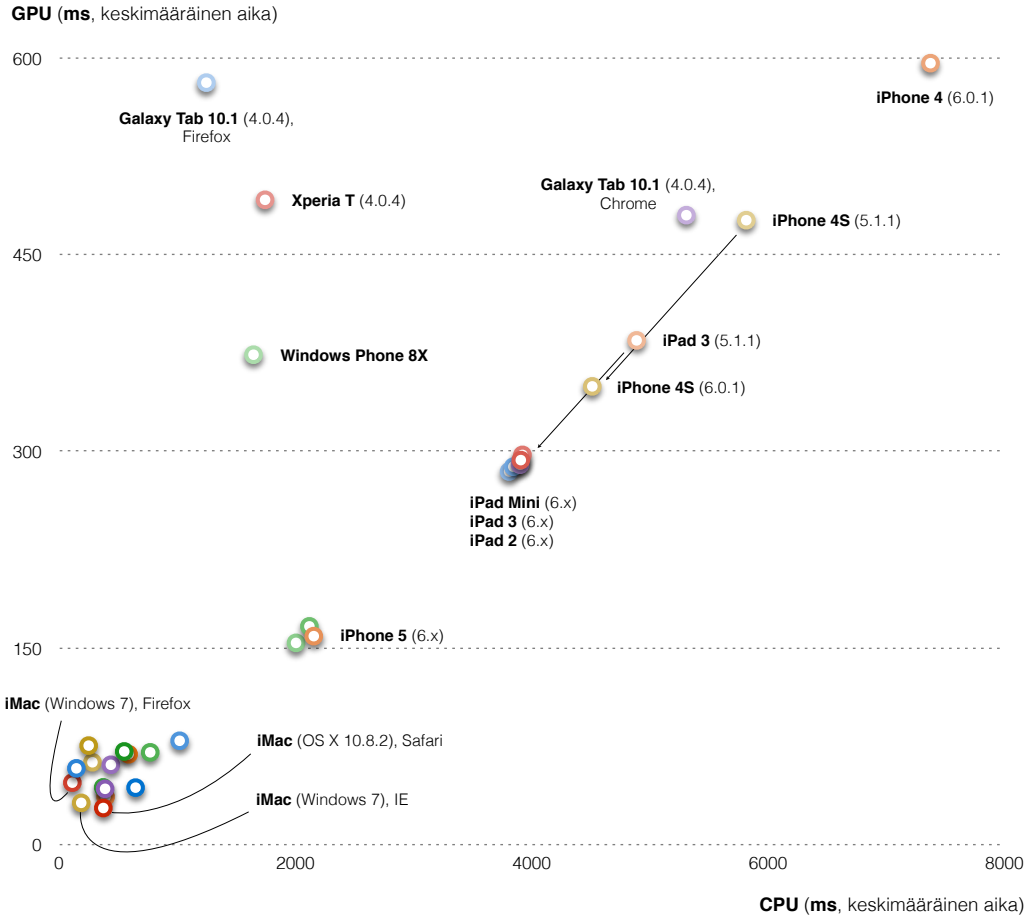
Galaxy S III ja Windows Phone 8X, jotka kuitenkin kamppailevat testin suoritusnopeudessa edellisen sukupolven iPhone 4S:n kanssa.



Kuva 6.13: CSS3-transformaatiotesti, mobiililaitteiden keskimääräinen suoritusaika. 95%-luottamusvälit merkitty vaaleammalla sinisellä.

Aikaisemmin esitelty ensimmäinen taustasuorittajatesti toimii hyvänä mittarina suorittimen suorituskyvyn hyödyntämiselle, joten sitä on käytetty yleispätevänä CPU-mittarina seuraavaksi esiteltäville kuvaajille. Vastaavasti CSS3-transformaatiotestin tulokset tarjoavat varteenotettavan mittarin laitteiden GPU-laskennalle. Kuvassa 6.14 on esitelty kuvaaja, jonka vaaka-akselille on merkitty ensimmäisen taustasuorittajatestin ja pystyakselille CSS3-transformaatiotestin keskimääräiset suoritusajat. Kuvaaja antaa viitteitä laitteiden suorituskyvyn painottumisesta CPU- tai GPU-intensiiviselle laskennalle. Kuvaajan perusteella on selvää, että testatut Xperia T ja Windows Phone 8X painottuvat Android Browser ja Internet Explorer

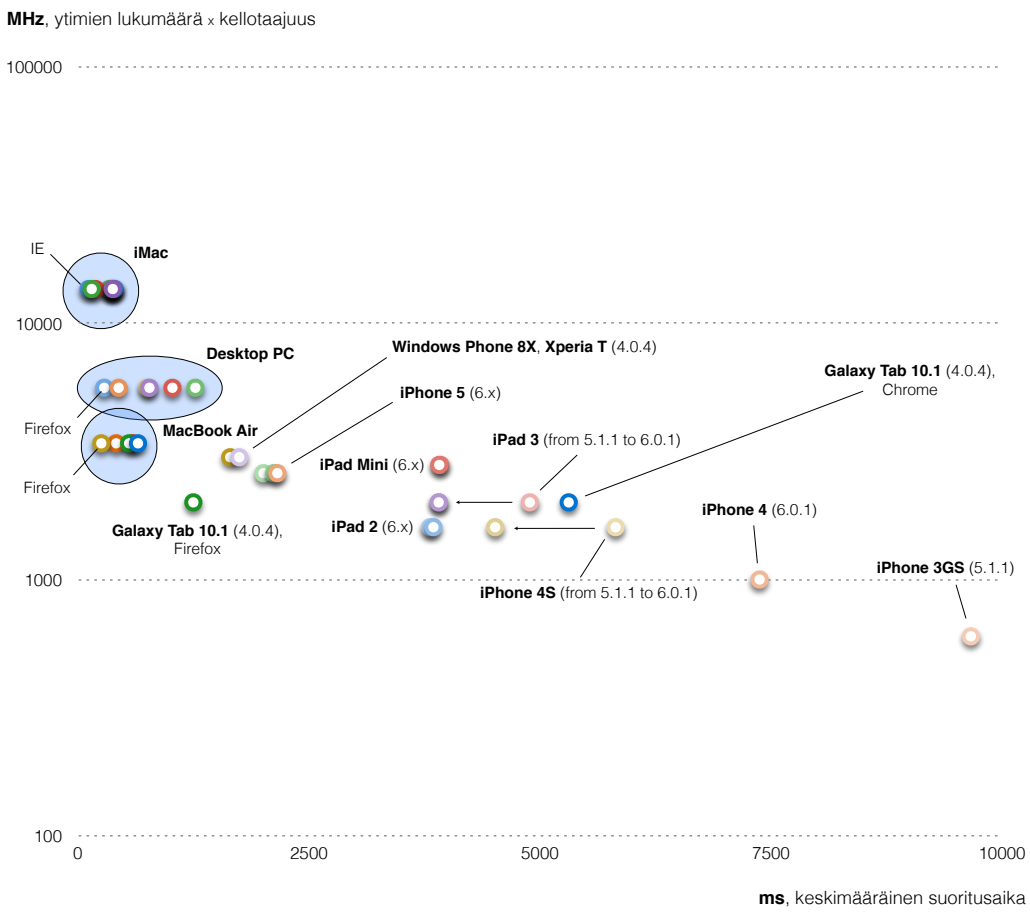
-selaimillaan suorittimella tapahtuvaan laskentaan. iOS-laitteet asettuvat tasaisesti kuvaajan keskilinjaan.



Kuva 6.14: Vaaka-akselilla ensimmäisen taustasuorittajatestin keskimääräinen suoritusaika millisekunteina, pystyakselilla CSS3-transformaatiotestin suoritusnopeus. Nopeimmat tulokset sekä CPU- että GPU-painotteisissa testeissä sijaitsevat kuvaajan vasemmassa alareunassa.

Kuvassa 6.14 nähdään myös kuinka iPhone 4S ja iPad 3 -laitteiden suorituskyky paranee tasaisesti eri ohjelmistoversioiden välillä sekä CPU- että GPU-intensiivisellä laskennalla mitattuna. Harppaus iPhone 4 ja iPhone 5 -laitteiden tulosten välillä on suuri nelinkertaistaen Applen lippulaivatuotteen suorituskyvyn reilussa kahdessa vuodessa. Työpöytäselaimista OS X:n Safari on parhaimmillaan graafisella suorituskyvyllä mitattuna, kun taas Windowsilla toimiva Firefox loistaa suorittimen hyödyntämisellä. Internet

Explorerin kymmenes versio näyttää tarjoavan tasapainotetun ratkaisun. Kuvassa 6.15 on esitetty vielä ensimmäisen taustasuorittajatestin tulokset verrattuna testatun laitteen suorittimen kellotaajuuteen kerrottuna suoritinydinten lukumäärällä.



Kuva 6.15: Vaaka-akselilla ensimmäisen taustasuorittajatestin keskimääräinen suoritusnopeus, pystyakselilla laitteen suorittimen ilmoitettu nopeus megahertseinä kerrottuna ytimien lukumäärällä.

Kuva 6.15 osoittaa ennen kaikkea ohjelmiston ja selainimplementaatioiden merkityksen suorituskyvyn kannalta. Paperilla heikommalla suorittimella varustettu Galaxy Tab pystyy Firefox-selaimen ansiosta lyömään testin suoritusnopeudessa monta kilpailevaa laitetta ja jättämään Chrome-selaimen kauas taakseen. Vastaavasti iPhone 3GS, iPhone 4 ja iPhone 5 -laitteiden testitulokset laittavat perspektiiviin mobiililaitteiden nopean kehityksen ja

tässä diplomityössä keskustellun asiakaslaitteiden konvergenssin.

6.3 Stage Framework

Stage Framework on diplomityön tuloksena syntynyt ohjelmistokehys, joka tarjoaa työkalut HTML5-muotoisten digitaalisten julkaisujen jakelemiseen web-applikaatiota käyttämällä. Ohjelmistokehys sisältää valmiin web-applikaation ja muut tarvittavat palvelinresurssit julkaisujen jakelemiseen. Se myös sisältää dokumentaation ohjelmistokehyksen käyttöönotosta, uuden julkaisun tekemisestä ja julkaisun numeroiden lisäämisestä.

Ohjelmistokehys on julkaistu vapaasti käytettäväksi ja se tarjoaa kokeellisen alustan julkaisujen tarjoamiseksi selaimessa toimivalla web-applikaatiolla hyödyntäen uusia HTML5- ja CSS3-teknologioita. Ohjelmistokehyksen web-applikaatio tukee WebKit-selainmoottorilla varustettuja selaimia, joten tabletti- ja mobiilialustoilla se toimii parhaiten iOS tai Android -mobiilikäyttöjärjestelmillä, joiden oletusselaimissa on WebKit. Puutteellisempi ohjelmistokehyksen tuki on tällä hetkellä Internet Explorer -selainta käyttäville alustoille, kuten Windows Phone -mobiilikäyttöjärjestelmille ja Windows RT:lle.

Ohjelmistokehyksen saa käyttöönsä lataamalla tarvittavat resurssit ohjelmistokehyksen verkkosivuilta[131]. Web-applikaation avulla tarjottavien julkaisujen resurssit sijoitetaan julkaisijan ylläpitämälle web-palvelimelle, jossa sijaitsee myös web-applikaation lehtihyllyn sisällön kuvaileva content.json-tiedosto. Tiedosto sisältää julkaisujen ja sen numeroiden tiedot JSON-muodossa ja sitä voidaan päivittää ohjelmallisesti tai manuaalisesti. Jokainen web-applikaatioon julkaistava numero tarvitsee myös kansikuvan kolmella eri tarkkuudella, jotka on määritelty ohjelmistokehyksen dokumentaatiossa. Ohjelmistokehys on saatavissa GPL- ja MIT-lisensseillä.

[131] `http://www.tml.tkk.fi/~ralatalo/stageframework/`

Luku 7

Analyysi ja tulkinta

Tässä luvussa analysoidaan ja vedetään yhteen diplomityö ja sen tulokset. Luvussa esitetään myös tulosten pohjalta koottu suositeltujen käytäntöjen lista digitaaliselle julkaisemiselle HTML5-tekniikalla, joka on myös itsessään diplomityössä syntynyt tulos. Kuten tulokset osiossa on esitelty, ovat toiset tutkituista teknologioista tällä hetkellä käyttökelpoisempia ja laajemmin tuettuja kuin toiset. Diplomityön analyysissa ja tulkinnassa on pyritty reflektoimaan myös tehtyjen toteutusratkaisujen onnistumista.

Yleisesti ottaen diplomityössä käsitellyt verrattain uudet HTML5- ja CSS3-tekniikat ovat osoittautuneet hyvin tuetuiksi, mikä tekee niistä erittäin käyttökelpoisia digitaalisessa julkaisemisessa. HTML5 on myös uudistetun semantiikkansa ansiosta luonteva kuvauskieli julkaisujen sisällölle. Kuten diplomityössä on aikaisemmin todettu, tarjoavat selain- ja JavaScript-moottorit lyömättömän suoritusympäristön: ne löytyvät jokaisesta tabletti- ja mobiililaitteesta, eivätkä vaadi käyttöönotettaessa ylimääräisiä toimenpiteitä käyttäjältä. Selaimella käytettävästä webistä on kasvanut avoin sovellusalusta, joka on kenen tahansa käytettävissä missä ja milloin vain [67].

Tabletti- ja mobiililaitteiden suorituskyky on kasvanut suuresti viimeisten vuosien aikana, mikä näkyy kaventuneena suorituskykyerona työpöytäjärjestelmiin. Tästä hyvä osoitus on iPhone 5 -puhelimen menestys suorituskykymittauksissa toteutetussa CSS3-transformaatiotestissä. iPhone 5:n testimenestys osoittaa erityisesti panostuksen laitteiden graafiseen suorituskykyyn, mutta myöskään taustasuorittajatesti ei jätä tulkinnanvaraa ARM-pohjaisten suorittimien suorituskyvyn parantumisesta ja moniytimisyyden tuomista hyödyistä.

Laitteiden suorituskyvyn kasvu ja web-teknologioiden kehittyminen ovat yhdessä tehneet web-applikaatioista varteenotettavamman toteutusratkaisun

natiivien sovellusten ja näköislehtien rinnalle. Esimerkiksi rautapohjaisen kiihdytyksen hyödyntäminen mahdollistaa näyttävien ja interaktiivisten sisältöjen luomisen pelkillä web-teknologioilla. Natiivit sovellukset ovat edelleen etulyöntiasemassa, mutta muun muassa kehitteillä olevat laiterajapinnat tulevat tulevaisuudessa kaventamaan etumatkaa entisestään.

Vuonna 2011 julkaistussa artikkelissaan [68] Taivalsaari ja Mikkonen vastasivat muun muassa Wired-lehden artikkelin The Web Is Dead [69] tarjoamaan näkökulmaan perinteisen Internetin kuolemasta, jossa yhtenä perusteena käytettiin natiivien sovellusten suosiota suhteessa perinteisiin selainpohjaisiin web-palveluihin. Taivalsaaren ja Mikkosen sanoin ei ollut olemassa perustavanlaatuisia teknisiä syitä miksi puhtaat web-applikaatiot eivät voisi yltää samaan kuin vastaavat natiivit sovellukset. Artikkelissa esiteltyjä mahdollistavia teknologioita, jotka vielä tuolloin olivat nuoria, on käsitelty ja sovellettu tässä diplomityössä menestyksekkäästi.

Responsiivisen suunnitteluperiaatteen hyödyntäminen HTML5-sisällössä mahdollistaa julkaisun ulkoasun vaivattomamman tuottamisen alusta- ja laiteriippumattomasti. HTML5-muotoinen sisältö myös mukautuu helpommin eri jakelukanaviin, joten samaa sisältöä voidaan hyödyntää esimerkiksi verkossa ja sovelluksien sisällä. Yksittäisistä teknologioista esimerkiksi web-kirjasimet helpottavat julkaisujen visuaalisen ilmeen hallintaa läpi jakelukanavien. Alusta- ja laiteriippumattomuus mahdollistaa resurssien kohdentamisen esimerkiksi uusien teknologioiden tuomien mahdollisuuksien hyödyntämiseen. Näitä ovat muun muassa audiovisuaalinen, auraalinen ja interaktiivinen sisältö sekä grafiikan, kuten infograafien animointi.

Diplomityössä toteutettujen sovellusten ja demolehden kehityksessä saatujen kokemusten perusteella koottiin suositeltujen käytäntöjen lista, joka ohjeistaa tutkittujen teknologioiden hyödyntämisessä. Käytännöt koskevat pääasiassa HTML5-sisältöä, mutta mukana on myös muutama vinkki sisällön jakeluun erityisesti web-applikaation näkökulmasta. Suositeltujen käytäntöjen lista on esitetty seuraavaksi.

Hyödynnä responsiivista suunnittelua ja CSS-mediakyselyitä

Responsiivisen ulkoasun hyödyntäminen tarkoittaa vähemmän työtä, joka kuluisi muuten laitekohtaiseen suunnitteluun. Laajasti tuetut uudet CSS-ominaisuudet mahdollistavat visuaalisesti näyttävien julkaisujen toteuttamisen. CSS-mediakyselyiden avulla sisältö saadaan optimoitua automaattisesti suuri laitevariaatio huomioiden. Responsiivinen suunnittelu ja CSS-mediakyselyt mahdollistavat myös tiedonsiirron optimoinnin tarjottaessa kullekin laitteelle sopiva kuvatarkkuus: tarkemmat ja suuremmat kuvat lait-

teille, jotka pystyvät ne hyödyntämään.

Mahdollista sisällön lataaminen laitteen muistiin

Erityisesti verkon yli sisältöä tarjottaessa olisi hyvä mahdollistaa sisällön pysyvä tai väliaikainen tallentaminen laitteen muistiin. Esimerkiksi julkaisun yhden numeron tallentaminen HTML5-sovellusvälimuistiin selaamisen ajaksi tuo merkittävän parannuksen suorituskykyyn ja sitä kautta käyttökokemukseen numeron sisältöä selattaessa. Myös muiden paikallisten tietokantaratkaisujen käytöllä voidaan vähentää tarvittavia HTTP-pyyntöjä. Laitteen muistin hyödyntäminen mahdollistaa myös offline-toiminnallisuuden tarjoamisen.

Animoi sisältöä CSS3-tekniikoilla, hyödynnä piirtoaluetta tarvittaessa

CSS3-siirtymät, -animaatiot ja -transformaatiot ovat laajasti tuettuja, joten niitä voidaan käyttää sisältöjen animointiin laite- ja alustariippumattomasti. CSS3 tarjoaa JavaScriptillä ja perinteisillä CSS-asetuksilla toteutettuja animointeja paremman suorituskyvyn. Se myös kestää HTML5-piirtoaluetta paremmin raskasta DOM-manipulaatiota ja muuta samanaikaisesti tapahtuvaa suorittimella tehtävää laskentaa. HTML5-piirtoalue on kuitenkin myös laajasti tuettu ja sitä voidaan käyttää resoluutioriippuvaisen grafiikan animointiin ja esittämiseen.

Hyödynnä rautapohjaista kiihdytystä, kun mahdollista

CSS3 mahdollistaa useilla alustoilla ja selaimilla rautapohjaisen kiihdytyksen hyödyntämisen. CSS3-siirtymät, -animaatiot ja -3D-transformaatiot voivat kaikki hyötyä näytönohjaimella tapahtuvasta laskennasta. Rautapohjaisen kiihdytyksen varmistamiseksi on myös olemassa alustakohtaisia käytäntöjä, paras keino on kuitenkin jo CSS3-ratkaisuihin tukeutuminen. Myös WebGL mahdollistaa rautapohjaisen kiihdytyksen, HTML5-piirtoaluetta käytettäessä.

Ehosta julkaisua web-kirjasimilla

Web-kirjaimet ovat tulleet jäädäkseen ja tarjoavat erinomaisen visuaalisen keinon pienellä resurssikuormalla. Suurin osa laitteista tukee vähintään joko perinteisiä TTF ja OTF -kirjasintyyppejä tai viimeistään uudeksi standardiksi kehitettyä WOFF-kirjasinta.

Suosi yleisimmin tuettuja, johtavia mediaformaatteja

HTML5:n mediaelementtien kanssa käytettäviä mediaformaatteja voi käytän-

nössä olla rajattomasti, sillä spesifikaatio ei niitä rajaa. Suurin osa Internetin videoista käyttää H.264/MPEG-4 AVC -formaattia, joten yksittäisen sisällöntuottajan näkökulmasta se on audioformaatti MP3:n kanssa luonnollisin valinta. Selaimista Firefoxin ja Operan työpöytäselaimet ovat viimeisiä linnakkeita, jotka eivät tue H.264-pakkausta oletuksena. Näihinkin selaimiin tuki on asennettavissa erillisellä pluginilla. Suurin osa laitteista tukee H.264-pakkauksen purkamista rautatasolla, joka tekee siitä suorituskykyisimmän verrattuna muihin pakkausformaatteihin.

Käytä SVG-grafiikkaa vapautuneesti

SVG on vektorigrafiikkaformaattina lyömätön kumppani responsiiviselle suunnittelulle ja digitaalisissa julkaisuissa tarvittaville visualisoinneille, kuten infograafeille. SVG on nykyään jo laajasti tuettu, joten sitä voi hyödyntää hyvällä luottamuksella. SVG mahdollistaa resoluutioriippuvaisten rasterigrafiikkaresurssien korvaamisen käyttöliittymäelementeissä.

Käytä edistyneempiä kirjasinasetuksia ja CSS-tavutusta harkiten

CSS-tavutus ja komannen tason kirjasinmoduulin edistyneet kirjasinasetukset ovat vielä rajallisesti tuettuja. Tekstien tavutuksen tueksi olisikin hyvä käyttää JavaScript-kirjastolla saavutettavaa tavutusta.

Suosi natiiveja selaintoteutuksia

Selaimessa toteutetut ominaisuudet ovat pääsääntöisesti aina tehokkaimmin toteutettuja. Esimerkkejä natiiveista selaintoteutuksista ovat muun muassa HTML5-piirtoalueen *toDataURL*-metodi, taustasuorittajat ja selaimen oma vieritys. Esimerkiksi vierityksen tapauksessa iOS-alustalla kannattaa hyödyntää kineettisen vierityksen erinomaista natiivia toteutusta ja säästää ominaisuuden replikointi muille alustoille.

Tunnista selainominaisuudet tarkoitukseen kehitetyllä kirjastolla

Selainten tukemia ominaisuuksia tunnistettaessa tulisi tukeutua yleisesti käytettyihin ja aktiivisesti ylläpidettyihin tunnistuskirjastoihin. Tunnistuskirjastot tarjoavat luotettavan ja konsistentin keinon teknologioiden tunnistamiseen. Esimerkiksi diplomityössä käytetty Modernizr tarjoaa myös sivun lataamisen yhteydessä lisättävät HTML-luokat eri teknologioille, mukaan lukien CSS3-tyyleille.

Hyödynnä taustasuorittajia niitä tukevissa selaimissa

Taustasuorittajat tarjoavat suorituskykyhyödyn rinnakkaisessa laskennassa

jo määrällisesti pienellä kuormalla. Diplomityössä taustasuorittajia hyödynnettiin kuvien base64-enkoodaukseen. Suorituskykyhyöty saavutettiin jo kolmen noin yhden megatavun kuvan rinnakkaisessa enkoodauksessa. Taustasuorittajat ovat verrattain hyvin tuettuja, erityisesti suorituskykyisemmissä laitteissa, joissa niiden hyödyntäminen kannattaa.

Suosi JSON-formaattia tiedonsiirrossa

JSON on kompakti tiedostoformaatti ja sen parsiminen on todella nopeaa kaikissa JavaScript-moottoreissa. Kevyen syntaksinsa vuoksi se on nopeammin käsiteltävissä ja tuotettavissa kuin esimerkiksi XML. JSONia voidaan käyttää tiedonsiirtoon myös eri domainien välillä AJAX-kutsuissa.

Edellä listatut suositellut käytännöt pyrkivät auttamaan HTML5-taitettujen julkaisujen toteutuksessa ja jakelussa. Käytäntöjä koostettaessa on hyödynnetty diplomityössä toteutettujen sovelluksien toteutusratkaisujen reflektointia. Esimerkiksi diplomityössä toteutetun web-applikaation selausnäkymän vieritys toteutettiin CSS3-transformaatiolla johtuen pääasiassa tekniikan hyvästä suorituskyvystä iOS-alustalla ja Android Browser -selainten puuttuvasta vieritystuesta ylijatkuville HTML-elementeille. Diplomityön aikana alustojen selainten implementaatiot ovat kuitenkin parantuneet. Lähtökohtaisesti selaimien omien toteutusten käyttö on aina suotavampaa parhaimman suorituskyvyn saavuttamiseksi, tästä syystä suositeltujen käytäntöjen listalla kannustetaankin niiden käyttöön.

Diplomityössä toteutetut sovellukset käyttävät molemmat samaa HTML5-taitettua demolehden sisältöä. Vertailtaessa natiivin iOS-wrapper-sovelluksen ja web-applikaation suorituskykyä, nousee merkittävimmäksi yksittäiseksi tekijäksi lehden resurssien sijainti. Web-applikaatiossa resurssit joudutaan lataamaan verkon ylitse käyttäjän selatessa lehteä, kun taas iOS-sovelluksessa lehti sijaitsee laitteessa paikallisesti. Lehden sivujen lataaminen on huomattavasti nopeampaa iOS-sovelluksessa kuin web-applikaatiossa. Diplomityössä on kuitenkin esitetty toimenpiteitä ja toteutusratkaisuja julkaisun suorituskyvyn parantamiseen verkon ylitse ladattaessa. Näihin kuuluvat muun muassa dynaaminen HTML5-sovellusvälimuisti.

Diplomityön aluksi esitettiin kolme tutkimuskysymystä:

- *Mikä on optimaalisin keino tarjoilla HTML5-taitettua sisältöä tabletti- ja mobiilialustoille?*
- *Millä keinoilla voidaan parantaa HTML5-tekniikalla julkaistujen lehtien suorituskykyä?*
- *Mitkä ovat keskeisiä ongelmia ja rajoitteita HTML5-julkaisemisessa tabletti- ja mobiilialustoille? Millä keinoilla näitä ongelmia voidaan ratkaista?*

Pohdittaessa optimaalisinta keinoa tarjoilla HTML5-taitettua sisältöä tabetti- ja mobiilialustoille, ovat merkittävässä roolissa julkaisijan omat vaatimukset ja rajoitteet, kuten tavoitellun levikin suhde käytettävissä oleviin resursseihin. Teknologiset valinnat vaikuttavat oleellisesti muun muassa tuetun ohjelmisto- ja laitekannan suuruuteen. Diplomityössä toteutetussa ohjelmistokehyksessä web-applikaation avulla tapahtuvaan HTML5-taitettujen lehtien jakeluun on pyritty soveltamaan niitä parhaita käytäntöjä, joita diplomityön aikana on identifioitu.

Toteutettujen alusta- ja selainkartoituksen sekä suorituskykymittausten tulosten valossa web-applikaatiototeutus on teknisesti hyvin tuettu eri alustoilla. Web-applikaatiototeutukset ovat myös jo arkipäivää, kuten diplomityössä käsitellyistä referenssijulkaisuistakin käy ilmi. Kuten tässäkin luvussa kuitenkin on todettu, saavutti toteutettu iOS-sovellus paremman suorituskyvyn erityisesti paikallisten sisältöresurssien ansiosta. Optimaalisen julkaisukeinon nimeäminen eri sovellus- ja julkaisutyyppien välillä ei ole tämän tutkimuksen valossa mielekästä, vaan jakelukeinon valinnassa on syytä huomioida eri toteutusratkaisujen vahvuudet tapauskohtaisesti. Esimerkiksi natiiveina sovelluksina julkaistujen lehtien paikallisten resurssien suuri koko voi koitua jopa julkaisujen kohtaloksi [70].

HTML5-tekniikalla julkaistujen lehtien suorituskykyä voidaan parantaa valittuun julkaisutyyppiin sopivilla avustavilla teknologioilla. Tässä diplomityössä on esitelty teknologioita erityisesti selainpohjaisten toteutusratkaisujen tueksi. Näitä teknologioita on myös sovellettu toteutetussa Stage Framework -ohjelmistokehyksessä. Keskeisinä rajoitteina HTML5-julkaisemisessa erityisesti selainpohjaisiin toteutuksiin tukeuduttaessa ovat olleet suorituskyky ja laitevariaatio. Tässä diplomityössä on kuitenkin käsitelty useita keinoja suorituskykyvn parantamiseksi käyttämällä teknologioita, jotka ovat mahdollisimman laajasti tuettuja.

Pohdittaessa käytettyjen teknologioiden ja digitaalisen julkaisemisen tulevaisuutta, ovat HTML5 ja siihen liittyvät teknologiat yhä merkittävämmässä roolissa. Natiivien sovellusten ja web-pohjaisten sovelluksien teknologiat lähentyvät entistä enemmän toisiaan tuottaen sovelluksien hybridimuotoja molempiin suuntiin. Esimerkiksi WebGL on hyvä esimerkki natiivien teknologioiden käytön mahdollistamisesta selaimissa. [71]

Steve Jobs kirjoitti vuonna 2010 avoimen kirjeen otsikolla *Ajatuksia Flash-tekniikasta* (engl. Thoughts on Flash) [72]. Kirjeessä Jobs ottaa kantaa muun muassa kritiikkiin Flash-tuen puuttumisesta iOS-alustalla. Jobs kannusti avoimien web-standardien, kuten HTML5:n käyttöön ja uskoi niiden voittavan Flashin kaltaiset suljetut tekniikat [72]. Vuonna 2010 kyse oli myös Adoben ja Applen välisestä taistelusta, mutta aika on osoittanut, että HTML5:stä on muuhunkin kuin suljettujen web-tekniikoiden haastajaksi: web-pohjaiset sovellukset pystyvät kamppailemaan tasaväkisesti natiivien sovellusten kanssa ja web-tekniikat ovat raivanneet tiensä pysyvästi myös natiiveihin sovellustoteutuksiin.

Luku 8

Lopuksi

Tässä diplomityössä on käsitelty digitaalisten aikakaus- ja sanomalehtien julkaisemista HTML5-tekniikalla. HTML5:n ja siihen liittyvien teknologioiden tuki on saavuttanut hyvän tason tabletti- ja mobiilialustoilla, mikä tekee niistä varteenotettavia digitaalisen julkaisemisen kannalta. Web-applikaatiot pystyvät saavuttamaan muun muassa natiivien sovellusten graafisen suorituskyvyn ja hyödyntämään tulevaisuudessa entistä paremmin laitteiden ominaisuuksia erilaisten laiterajapintojen avulla. Selaimessa toimivien web-pohjaisilla teknologioilla toteutettujen sovellusten ja julkaisujen tulevaisuus näyttää erittäin lupaavalta, mutta jakelukanava on hyödynnettävissä jo nyt.

Tabletti- ja mobiililaitteiden nopea kehittyminen on palvellut maailmaa monella tapaa: arkisiin tehtäviin, kuten verkkopalveluiden käyttämiseen tarvitaan ARM-pohjaisilla laitteilla enää murto-osa energiaa verrattuna perinteisiin x86-pohjaisiin pöytäkoneisiin ja kannettaviin tietokoneisiin. Samaan aikaan langattomien yhteyksien kehittyminen on mahdollistanut mukana kulkevien laitteiden täysimittaisen hyödyntämisen ja vahvistanut Internetin roolia ihmisten arkielämässä. Tämä teknologioiden konvergenssi nostaa selainpohjaisten sovellusten, sisältöjen ja palveluiden merkitystä, jotka voivat kulkea eri laitteiden välillä ja olla käyttäjien saavutettavissa ajasta, paikasta ja laitteesta riippumatta.

Tämän diplomityön menetelmillä saatuja tuloksia voidaan soveltaa tulevaisuudessa tutkittujen teknologioiden kehittyessä kumulatiivisesti, ne pysyvät siis ajankohtaisina vielä pitkään. Tästä huolimatta esimerkiksi toteutetut alusta- ja selainkartoitus sekä suorituskykymittaukset vanhenevat sellaisinaan yhtä nopeasti kuin uusia laitteita, ohjelmistoversioita ja selainteknologioita kehitetään. Näiden kvantitatiivisten menetelmien tarkoituksena on kuitenkin ollut rohkaista digitaalisten julkaisujen toteuttajia ja web-kehittäjiä hyödyntämään uusia teknologioita rohkeasti, sillä ne ovat enemmän kuin käytettävissä: niillä voidaan parantaa digitaalisten julkaisujen käyttökokemusta, rikastaa sisältöä sekä toteuttaa visuaalisesti näyttäviä ja pikselintarkkoja ilmeitä käyttäen aitoja, tehtävään suunniteltuja, tarkoituksenmukaisia teknologioita.

Lähteet

[1] Kuhna, M., Kivelä, I. M., Oittinen, P., "Semi-Automated Magazine Layout Using Content-based Image Features," *Proceedings of the 20th ACM international conference on Multimedia*, MM '12, (New York, NY, USA), pp. 379–388, ACM, 2012. ISBN: 9781450310895, DOI: 10.1145/2393347.2393403.

[2] Gupta, A., Milanesi, C., Cozza, R., Zimmermann, A., Lu, C. K., Nguyen, T. H., De La Vergne, H. J., Glenn, D., "Market Share: Mobile Phones by Region and Country, 3Q12," 13. marraskuuta 2012. Saatavilla: `http://www.gartner.com/resId=2236115`. Viitattu 7. tammikuuta 2013.

[3] Gupta, A., Cozza, R., Milanesi, C., Lu, C. K., "Market Share Analysis: Mobile Phones, Worldwide, 4Q12 and 2012," 12. helmikuuta 2012. Saatavilla: `http://www.gartner.com/resId=2334916`. Viitattu 20. helmikuuta 2013.

[4] Firtman, M., *jQuery Mobile: Up and Running*. O'Reilly Media, Inc., 2012. 254 s. ISBN: 1449397654, 9781449397654.

[5] Wynne, P., *Pimp My Site: The DIY Guide to SEO, Search Marketing, Social Media and Online PR*. John Wiley Sons, 2011. 274 s. ISBN: 0857082426, 9780857082428.

[6] Baghdassarian, S., Frank, A., "Forecast: Mobile Advertising, Worldwide, 2009-2016," 21. marraskuuta 2012. Saatavilla: `http://www.gartner.com/resId=2247015`. Viitattu 14. tammikuuta 2013.

[7] Sathayan, J., Anoop, N., Narayan, N., Shibu, K. V., *A Comprehensive Guide to Enterprise Mobility*. CRC Press, 2012. 556 s. ISBN: 1439867356, 9781439867358.

[8] McCann, T., *The Art of the App Store: The Business of Apple Development*. John Wiley Sons, 2011. 336 s. ISBN: 1118235347, 9781118235348.

[9] Restivo, K., Llamas, R., Shirer, M., "Strong Demand for Smartphones and Heated Vendor Competition Characterize the Worldwide Mobile Phone Market at the End of 2012, IDC Says," 24. tammikuuta 2013. Saatavilla: `http://www.idc.com/getdoc.jsp?containerId=prUS23916413`. Viitattu 8. helmikuuta 2013.

[10] Cortimiglia, M., Ghezzi, A., Renga, F., "Mobile Applications and Their Delivery Platforms," *IT Professional*, vol. 13, pp. 51–56, syyskuu 2011. DOI: 10.1109/MITP.2011.84.

[11] David, M., *HTML5 Mobile Websites: Turbocharging HTML5 with jQuery, Sencha Touch, and Other Frameworks*. Elsevier Science Technology Books, 2011. 236 s. ISBN: 024081813X, 9780240818139.

[12] Berjon, R., Leithead, T., Navara, E. D., O'Connor, E., Pfeiffer, S., Hickson, I., "HTML5 - A vocabulary and associated APIs for HTML and XHTML. W3C Candidate Recommendation 17 December 2012," Candidate Recommendation, W3C, 17. joulukuuta 2012. Saatavilla: `http://www.w3.org/TR/2012/CR-html5-20121217/`. Viitattu 24. tammikuuta 2013.

[13] Cabanier, R., Graff, E., Munro, J., Wiltzius, T., Hickson, I., "HTML Canvas 2D Context, Level 2. W3C Working Draft 17 December 2012," Working Draft, W3C, 17. joulukuuta 2012. Saatavilla: `http://www.w3.org/TR/2012/WD-2dcontext2-20121217/`. Viitattu 8. helmikuuta 2013.

[14] Hickson, I., "Web Storage. W3C Candidate Recommendation 08 December 2011," Recommendation, W3C, 8. joulukuuta 2011. Saatavilla: `http://www.w3.org/TR/2011/CR-webstorage-20111208/`. Viitattu 8. helmikuuta 2013.

[15] Aamulehto, R., "Feature Detection Results - Digital Magazine and Newspaper Publishing with HTML5." Tulosdokumentti, 2013. Saatavilla: `http://www.tml.tkk.fi/~ralatalo/FDR.pdf`.

[16] Irish, P., "Best Practices for a Faster Web App with HTML5 - HTML5 Rocks." Ohjedokumentti, 18. kesäkuuta 2010. Saatavilla: `http://www.html5rocks.com/en/tutorials/speed/quick/`. Viitattu 8. helmikuuta 2013.

[17] Berjon, R., Leithead, T., Navara, E. D., O'Connor, E., Pfeiffer, S., Hickson, I., "HTML 5.1 - A vocabulary and associated APIs for HTML and XHTML. W3C Working Draft 17 December 2012," Working Draft, W3C, 17. joulukuuta 2012. Saatavilla: `http://www.w3.org/TR/2012/WD-html51-20121217/`. Viitattu 24. tammikuuta 2013.

[18] Harjono, J., Ng, G., Kong, D., Lo, J., "Building smarter web applications with HTML5," *Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '10, (Riverton, NJ, USA), pp. 402–403, IBM Corp., 2011. DOI: 10.1145/1923947.1924015.

[19] Etemad, E. J., "Cascading Style Sheets (CSS) Snapshot 2010. W3C Working Group Note 12 May 2011," Working Group Note, W3C, 12. toukokuuta 2011. Saatavilla: `http://www.w3.org/TR/2011/NOTE-css-2010-20110512/`. Viitattu 8. helmikuuta 2013.

[20] Çelik, T., Etemad, E. J., Glazman, D., Hickson, I., Linss, P., Williams, J., "Selectors Level 3. W3C Recommendation 29 September 2011," Recommendation, W3C, 29. syyskuuta 2011. Saatavilla: `http://www.w3.org/TR/2011/REC-css3-selectors-20110929/`. Viitattu 8. helmikuuta 2013.

[21] Rivoal, F., Lie, H. W., Çelik, T., Glazman, D., van Kesteren, A., "Media Queries. W3C Recommendation 19 June 2012," Recommendation, W3C, 19. kesäkuuta 2012. Saatavilla: `http://www.w3.org/TR/2012/REC-css3-mediaqueries-20120619/`. Viitattu 8. helmikuuta 2013.

[22] Çelik, T., Lilley, C., Baron, L. D., Pemberton, S., Pettit, B., "CSS Color Module Level 3. W3C Recommendation 07 June 2011," Recommendation, W3C, 7. kesäkuuta 2011. Saatavilla: `http://www.w3.org/TR/2011/REC-css3-color-20110607/`. Viitattu 8. helmikuuta 2013.

[23] Daggett, J., "CSS Fonts Module Level 3. W3C Working Draft 11 December 2012," Working Draft, W3C, 11. joulukuuta 2012. Saatavilla: `http://www.w3.org/TR/2012/WD-css3-fonts-20121211/`. Viitattu 8. helmikuuta 2013.

[24] Fraser, S., Jackson, D., O'Connor, E., Schulze, D., Gregor, A., "CSS Transforms. W3C Working Draft 11 September 2012," Working Draft, W3C, 11. syyskuuta 2012. Saatavilla: `http://www.w3.org/TR/2012/WD-css3-transforms-20120911/`. Viitattu 8. helmikuuta 2013.

[25] Jackson, D., Hyatt, D., Marrin, C., Baron, L. D., "CSS Transitions. W3C Working Draft 3 April 2012," Working Draft, W3C, 3. huhtikuuta 2012. Saatavilla: `http://www.w3.org/TR/2012/WD-css3-transitions-20120403/`. Viitattu 8. helmikuuta 2013.

[26] Jackson, D., Hyatt, D., Marrin, C., Galineau, S., Baron, L. D., "CSS Animations. W3C Working Draft 3 April 2012," Working Draft, W3C, 3. huhtikuuta 2012. Saatavilla: `http://www.w3.org/TR/2012/WD-css3-animations-20120403/`. Viitattu 8. helmikuuta 2013.

[27] Kool, M., "Let's Play With Hardware-Accelerated CSS." Ohjedokumentti, 21. kesäkuuta 2012. Saatavilla: `http://mobile.smashingmagazine.com/2012/06/21/play-with-hardware-accelerated-css/`. Viitattu 9. helmikuuta 2013.

[28] Dahlström, E., Dengler, P., Grasso, A., Lilley, C., McCormack, C., Schepers, D., Watt, J., Ferraiolo, J., Jun, F., Jackson, D., "Scalable Vector Graphics (SVG) 1.1 (Second Edition). W3C Recommendation 16 August 2011," Recommendation, W3C, 16. elokuuta 2011. Saatavilla: `http://www.w3.org/TR/2011/REC-SVG11-20110816/`. Viitattu 8. helmikuuta 2013.

[29] Hickson, I., "Web Workers. W3C Candidate Recommendation 01 May 2012," Candidate Recommendation, W3C, 1. toukokuuta 2012. Saatavilla: `http://www.w3.org/TR/2012/CR-workers-20120501/`. Viitattu 8. helmikuuta 2013.

[30] Schepers, D., Moon, S., Brubeck, M., "Touch Events version 1. W3C Working Draft 24 January 2013," Working Draft, W3C, 24. tammikuuta 2013. Saatavilla: `http://www.w3.org/TR/2013/WD-touch-events-20130124/`. Viitattu 8. helmikuuta 2013.

[31] Hickson, I., "The WebSocket API. W3C Candidate Recommendation 20 September 2012," Candidate Recommendation, W3C, 20. syyskuuta 2012. Saatavilla: `http://www.w3.org/TR/2012/CR-websockets-20120920/`. Viitattu 8. helmikuuta 2013.

[32] Bergkvist, A., Burnett, D. C., Jennings, C., Narayanan, A., "WebRTC 1.0: Real-time Communication Between Browsers. W3C Working Draft 21 August 2012," Working Draft, W3C, 21. elokuuta 2012. Saatavilla: `http://www.w3.org/TR/2012/WD-webrtc-20120821/`. Viitattu 8. helmikuuta 2013.

[33] Marrin, C., "WebGL Specification. Editor's Draft 25 January 2013," Editor's Draft, Khronos Group, 25. tammikuuta 2013. Saatavilla: `http://www.khronos.org/registry/webgl/specs/latest/`. Viitattu 8. helmikuuta 2013.

[34] Ranganathan, A., Sicking, J., "File API. W3C Working Draft 25 October 2012," Working Draft, W3C, 25. lokakuuta 2012. Saatavilla: `http://www.w3.org/TR/2012/WD-FileAPI-20121025/`. Viitattu 8. helmikuuta 2013.

[35] Block, S., Popescu, A., "DeviceOrientation Event Specification. W3C Working Draft 1 December 2011," Working Draft,

W3C, 1. joulukuuta 2011. Saatavilla: `http://www.w3.org/TR/2011/WD-orientation-event-20111201/`. Viitattu 17. helmikuuta 2013.

[36] Marcotte, E., *Responsive Web Design*. Editions Eyrolles, 2011. 150 s. ISBN: 2212133316, 9782212133318.

[37] Smashing Magazine, *Responsive Design*. Mashing Magazine, 2012. 159 s. ISBN: 3943075338, 9783943075335.

[38] Frain, B., *Responsive Web Design with HTML5 and CSS3*. Packt Publishing Ltd, 2012. 305 s. ISBN: 1849693188, 9781849693189.

[39] Gillenwater, Z. M., *Flexible Web Design: Creating Liquid and Elastic Layouts with CSS*. Peachpit Press, 2010. 336 s. ISBN: 0132104687, 9780132104685.

[40] Gustafson, A., Zeldman, J., *Adaptive Web Design: Crafting Rich Experiences with Progressive Enhancement*. Easy! Designs, LLC, 2011. 144 s. ISBN: 098358950X, 9780983589501.

[41] Fling, B., *Mobile Design and Development: Practical Techniques for Creating Mobile Sites and Web Apps*. O'Reilly Media, Inc., 2009. 336 s. ISBN: 1449379249, 9781449379247.

[42] Mahemoff, M., *Ajax Design Patterns*. O'Reilly Media, Inc., 2009. 656 s. ISBN: 0596553617, 9780596553616.

[43] Mikowski, M., Powell, J., *Single Page Web Applications: JavaScript End-To-End*. Manning Publications Company, 2013. 325 s. ISBN: 1617290750, 9781617290756.

[44] Leff, A., Rayfield, J. T., "Web-Application Development Using the Model/View/Controller Design Pattern," *Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing*, EDOC '01, (Washington, DC, USA), pp. 118–, IEEE Computer Society, 2001. ISBN: 076951345X, DOI: 10.1109/EDOC.2001.950428.

[45] Puputti, K., "Mobile HTML5: Implementing a Responsive Cross-Platform Application." Diplomityö, Aalto-yliopiston perustieteiden korkeakoulu, 2012. Saatavilla: `http://kpuputti.github.com/thesis/files/thesis.pdf`. Viitattu 24. tammikuuta 2013.

[46] Souders, S., *High Performance Web Sites: Essential Knowledge for Front-End Engineers*. O'Reilly Media, Inc., 2008. 170 s. ISBN: 0596550693, 9780596550691.

[47] Souders, S., *Even Faster Web Sites*. O'Reilly Media, Inc., 2009. 256 s. ISBN: 0596555849, 9780596555849.

[48] Crockford, D., *JavaScript: The Good Parts*. O'Reilly Media, Inc., 2008. 172 s. ISBN: 0596554877, 9780596554873.

[49] Pilgrim, M., *HTML5: Up And Running*. O'Reilly Media, Inc., 2010. 222 s. ISBN: 1449399665, 9781449399665.

[50] The Financial Times Ltd, "FT Web App." Web-applikaatio, 2013. Saatavilla: `http://app.ft.com/`. Viitattu 8. helmikuuta 2013.

[51] The New York Times Company, "The Boston Globe." Verkkosivusto, 2013. Saatavilla: `http://bostonglobe.com/`. Viitattu 8. helmikuuta 2013.

[52] Apple Inc., Sanoma News Oy, "HS - Helsingin Sanomat for iPhone, iPod touch and iPad on the iTunes App Store." iOS-sovellus, 2013. Saatavilla: `https://itunes.apple.com/fi/app/hs-helsingin-sanomat/id380643421`. Viitattu 8. helmikuuta 2013.

[53] Sanoma News Oy, "Päivän lehti - Helsingin Sanomat." Web-applikaatio, 2013. Saatavilla: `http://hs.fi/paivanlehti/`. Viitattu 8. helmikuuta 2013.

[54] Apple Inc., Otavamedia Oy, "Suomen Kuvalehti - Uusi Versio for iPhone, iPod touch and iPad on the iTunes App Store." iOS-sovellus, 2013. Saatavilla: `https://itunes.apple.com/fi/app/suomen-kuvalehti-uusi-versio/id496510502`. Viitattu 8. helmikuuta 2013.

[55] Otavamedia Oy, "SK digi." Web-applikaatio, 2013. Saatavilla: `http://digi.suomenkuvalehti.fi/`. Viitattu 8. helmikuuta 2013.

[56] Engelhardt, N., Ippen, J., "Aside Magazine." Web-applikaatio, 2013. Saatavilla: `http://asidemag.com/`. Viitattu 8. helmikuuta 2013.

[57] Kivelä, I. M., "Aesthetic measures for automated magazine layout on tablet devices." Diplomityö, Aalto-yliopiston perustieteiden korkeakoulu, 2012. Saatavilla: `http://media.tkk.fi/visualmedia/publications/msc-theses/DI_I-M_Kivela_2012.pdf`. Viitattu 24. tammikuuta 2013.

[58] Pekkala, S., "Usability evaluation of design solutions for tablet magazines." Diplomityö, Aalto-yliopiston perustieteiden korkeakoulu, 2012. Saatavilla: `http://media.tkk.fi/visualmedia/publications/msc-theses/DI_S_Pekkala_2012.pdf`. Viitattu 24. tammikuuta 2013.

[59] Hiilivirta, J., "Aalto University tablet magazine -konseptikäsikirja." Aalto-yliopiston kurssin 24265 Publication Design -kurssityö, 2012. Saatavilla: `http://www.tml.tkk.fi/~ralatalo/AUM-Tablet.pdf`. Viitattu 22. helmikuuta 2013.

[60] Aamulehto, R., "Performance Measurement Results - Digital Magazine and Newspaper Publishing with HTML5." Tulosdokumentti, 2013. Saatavilla: `http://www.tml.tkk.fi/~ralatalo/PMR.pdf`.

[61] Keinänen, L. M., Haaramo, M., "Aalto University Magazine iPad-sovellus. Käytettävyyden asiantuntija-arviointi." , 11. lokakuuta 2012.

[62] Aamulehto, R., "Stage Framework." Web-applikaatio, 2013. Saatavilla: `http://www.tml.tkk.fi/~ralatalo/stage/`. Viitattu 8. helmikuuta 2013.

[63] Saffer, D., *Designing Gestural Interfaces*. O'Reilly Media, Inc., 2008. 272 s. ISBN: 0596554222, 9780596554224.

[64] Barker, T., *Pro JavaScript Performance: Monitoring and Visualization*. Apress, 2012. 275 s. ISBN: 1430247495, 9781430247494.

[65] Mann, J., Wang, Z., Quach, A., "User Timing, Editor's Draft July 11, 2012," Editor's Draft, W3C, 11. heinäkuuta 2012. Saatavilla: `https://dvcs.w3.org/hg/webperf/raw-file/tip/specs/UserTiming/Overview.html`. Viitattu 17. helmikuuta 2013.

[66] Anttonen, M., Salminen, A., Mikkonen, T., Taivalsaari, A., "Transforming the web into a real application platform: new technologies, emerging trends and missing pieces," *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, (New York, NY, USA), pp. 800–807, ACM, 2011. ISBN: 9781450301138, DOI: 10.1145/1982185.1982357.

[67] Jacobs, I., Jaffe, J., Hegaret, P. L., "How the Open Web Platform Is Transforming Industry," *Internet Computing, IEEE*, vol. 16, pp. 82–86, marras-joulukuu 2012. DOI: 10.1109/MIC.2012.134.

[68] Taivalsaari, A., Mikkonen, T., "The Web as an Application Platform: The Saga Continues," *Proceedings of the 2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, SEAA '11, (Washington, DC, USA), pp. 170–174, IEEE Computer Society, 2011. ISBN: 9780769544885, DOI: 10.1109/SEAA.2011.35.

[69] Anderson, C., Wolff, M., "The Web is Dead: Long Live the Internet," *Wired*, vol. 18, pp. 118–127, 164–166, syyskuu 2010.

[70] Siegler, M. G., "Why Magazine Apps Suck." Kirjoitus, artikkeli, joulukuu 2012. Saatavilla: `http://techcrunch.com/2012/12/04/the-dakly-died-of-suckage/`. Viitattu 28. helmikuuta 2013.

[71] Charland, A., Leroux, B., "Mobile Application Development: Web vs. Native," *Communications of the ACM*, vol. 54, pp. 49–53, toukokuu 2011. DOI: 10.1145/1941487.1941504.

[72] Jobs, S., "Thoughts on Flash." Kirjoitus, avoin kirje, huhtikuu 2010. Saatavilla: `http://www.apple.com/hotnews/thoughts-on-flash/`. Viitattu 17. helmikuuta 2013.

Liite A

Testatut laitteet ja selaimet

Kannettavat ja työpöytäkoneet (x86)

Tunniste	Käyttöjärjestelmä	Selain	Selaimen versio
MacBook Air, Safari	OS X 10.8.2	Safari	6.0.2 (8536.26.17)
MacBook Air, Chrome	OS X 10.8.2	Chrome	23.0.1271.97
MacBook Air, Firefox	OS X 10.8.2	Firefox	16.0.2
MacBook Air, Opera	OS X 10.8.2	Opera	12.10 build 1652
MacBook Air, Maxthon	OS X 10.8.2	Maxthon	1.0.4.2000 Beta
iMac (OS X 10.8.2), Safari	OS X 10.8.2	Safari	6.0.2 (8536.26.17)
iMac (OS X 10.8.2), Chrome	OS X 10.8.2	Chrome	24.0.1312.56
iMac (OS X 10.8.2), Firefox	OS X 10.8.2	Firefox	18.0
iMac (OS X 10.8.2), Opera	OS X 10.8.2	Opera	12.12 build 1707
iMac (OS X 10.8.2), Maxthon	OS X 10.8.2	Maxthon	4.0.3.1000 RC
iMac (Windows 7), IE	Windows 7 SP1	Internet Explorer	10.0.9200.16438
iMac (Windows 7), Chrome	Windows 7 SP1	Chrome	24.0.1312.56 m
iMac (Windows 7), Firefox	Windows 7 SP1	Firefox	18.0
iMac (Windows 7), Opera	Windows 7 SP1	Opera	12.12 build 1707
iMac (Windows 7), Maxthon	Windows 7 SP1	Maxthon	4.0.0.2000 (prev.)
Desktop PC, IE	Windows 7 SP1	Internet Explorer	10.0.9200.16438
Desktop PC, Safari	Windows 7 SP1	Safari	5.1.7 (7534.57.2)
Desktop PC, Chrome	Windows 7 SP1	Chrome	23.0.1271.97 m
Desktop PC, Firefox	Windows 7 SP1	Firefox	16.0.2
Desktop PC, Opera	Windows 7 SP1	Opera	12.10 build 1652
Desktop PC, Maxthon	Windows 7 SP1	Maxthon	4.0.0.2000 (prev.)

MacBook Air viittaa 13-tuumaiseen, 2011-vuoden puolivälin Applen kannettavaan tietokoneeseen kaksiytimisellä 1,7GHz Intel Core i5 -suorittimella, 4GB 1333MHz DDR3 -muistilla, Intel HD Graphics 3000 384MB -näytönohjaimella ja 128GB SSD -kiintolevyllä.

iMac viittaa 27-tuumaiseen, 2012-vuoden lopun Applen työpöytäkoneeseen neliytimisellä 3,4GHz Intel Core i7 -suorittimella, 8GB 1600MHz DDR3 -muistilla, NVIDIA GeForce GTX 680MX 2048MB -

näytönohjaimella ja 1TB Fusion Drive -kiintolevyllä.

Desktop PC viittaa AMD-pohjaiseen PC-kokoonpanoon kaksiytimisellä 2,8GHz AMD Athlon X2 7850 Black Edition -suorittimella, 6GB 800MHz DDR2 -muistilla, NVIDIA GeForce GTX 260 896MB -näytönohjaimella ja 750GB HDD -kiintolevyllä.

Tabletti- ja mobiililaitteet (ARM)

Tunniste	**Käyttöjärjestelmä**	**Selain**	**Selaimen v.**
iPhone 5 (6.1)	iOS 6.1	Safari	*
iPad Mini (6.1)	iOS 6.1	Safari	*
iPad 3 (6.1)	iOS 6.1	Safari	*
iPad 2 (6.1)	iOS 6.1	Safari	*
iPhone 5 (6.0.2)	iOS 6.0.2	Safari	*
iPhone 4S (6.0.1)	iOS 6.0.1	Safari	*
iPhone 4 (6.0.1)	iOS 6.0.1	Safari	*
iPad Mini (6.0.1)	iOS 6.0.1	Safari	*
iPad 3 (6.0.1)	iOS 6.0.1	Safari	*
iPad 2 (6.0.1)	iOS 6.0.1	Safari	*
iPhone 5 (6.0)	iOS 6.0	Safari	*
iPhone 4S (5.1.1)	iOS 5.1.1	Safari	*
iPhone 3GS (5.1.1)	iOS 5.1.1	Safari	*
iPad 3 (5.1.1)	iOS 5.1.1	Safari	*
Galaxy S III (4.0.4)	Android 4.0.4	Android Browser	*
Xperia T (4.0.4)	Android 4.0.4	Android Browser	*
One S (4.0.3)	Android 4.0.3	Android Browser	*
One X (4.0.3)	Android 4.0.3	Android Browser	*
One V (4.0.3)	Android 4.0.3	Android Browser	*
C8812 (4.0.3)	Android 4.0.3	Android Browser	*
Xoom (4.0.4)	Android 4.0.4	Android Browser	*
Galaxy Tab 10.1 (4.0.4), Android Browser	Android 4.0.4	Android Browser	*
Galaxy Tab 10.1 (4.0.4), Chrome	Android 4.0.4	Chrome	18.0.1025469
Galaxy Tab 10.1 (4.0.4), Firefox	Android 4.0.4	Firefox	18.0
Galaxy Tab 10.1 (4.0.4), Opera Mobile	Android 4.0.4	Opera Mobile	12.1.3
Galaxy Tab 10.1 (4.0.4), Dolphin	Android 4.0.4	Dolphin Browser	9.2.0
Galaxy Tab (2.3.6)	Android 2.3.6	Android Browser	*
Windows Phone 8X (8)	Windows Phone 8	IE Mobile	10.0
Lumia 900 (7.5)	Windows Phone 7.5	IE Mobile	9.0

* Käyttöjärjestelmä määrää selaimen version.

Liite B

Tuetut kirjasintyypit ja mediaformaatit

Kannettavat ja työpöytäkoneet (x86), kirjasintyypit

Tunniste	TTF/OTF	WOFF	SVG	EOT
MacBook Air, Safari	✓	✓	✓	✗
MacBook Air, Chrome	✓	✓	✓	✗
MacBook Air, Firefox	✓	✓	✗	✗
MacBook Air, Opera	✓	✓	✗	✗
MacBook Air, Maxthon	✓	✓	✓	✗
Desktop PC, IE	✗	✓	✗	✓
Desktop PC, Safari	✓	✓	✓	✗
Desktop PC, Chrome	✓	✓	✓	✗
Desktop PC, Firefox	✓	✓	✗	✗
Desktop PC, Opera	✓	✓	✗	✗
Desktop PC, Maxthon	✓	✓	✓	✗

Tabletti- ja mobiililaitteet (ARM), kirjasintyypit

Tunniste	**TTF/OTF**	**WOFF**	**SVG**	**EOT**
iPhone 5 (6.1)	✓	✓	✓	✗
iPad Mini (6.1)	✓	✓	✓	✗
iPad 3 (6.1)	✓	✓	✓	✗
iPad 2 (6.1)	✓	✓	✓	✗
iPhone 5 (6.0.2)	✓	✓	✓	✗
iPhone 4S (6.0.1)	✓	✓	✓	✗
iPhone 4 (6.0.1)	✓	✓	✓	✗
iPad Mini (6.0.1)	✓	✓	✓	✗
iPad 3 (6.0.1)	✓	✓	✓	✗
iPad 2 (6.0.1)	✓	✓	✓	✗
iPhone 5 (6.0)	✓	✓	✓	✗
iPhone 4S (5.1.1)	✓	✓	✓	✗
iPhone 3GS (5.1.1)	✓	✓	✓	✗
iPad 3 (5.1.1)	✓	✓	✓	✗
Galaxy S III (4.0.4)	✓	✓	✓	✗
Xperia T (4.0.4)	✓	✓	✓	✗
One S (4.0.3)	✓	✗	✓	✗
One X (4.0.3)	✓	✗	✓	✗
One V (4.0.3)	✓	✗	✓	✗
C8812 (4.0.3)	✓	✗	✓	✗
Xoom (4.0.4)	✓	✗	✓	✗
Galaxy Tab 10.1 (4.0.4), Android Browser	✓	✓	✓	✗
Galaxy Tab 10.1 (4.0.4), Chrome	✓	✓	✓	✗
Galaxy Tab 10.1 (4.0.4), Firefox	✓	✓	✗	✗
Galaxy Tab 10.1 (4.0.4), Opera Mobile	✓	✓	✗	✗
Galaxy Tab 10.1 (4.0.4), Dolphin	✓	✓	✓	✗
Galaxy Tab (2.3.6)	✓	✗	✗	✗
Windows Phone 8X (8)	✗	✓	✗	✗
Lumia 900 (7.5)	✗	✗	✗	✗

Kannettavat ja työpöytäkoneet (x86), audioformaatit

Tunniste	**MP3**	**M4A**	**Ogg**	**WAV**
MacBook Air, Safari	✓	✓	✗	✓
MacBook Air, Chrome	✓	✓	✓	✓
MacBook Air, Firefox	✗	✗	✓	✓
MacBook Air, Opera	✗	✗	✓	✓
MacBook Air, Maxthon	✓	✓	✓	✓
Desktop PC, IE	✓	✓	✗	✗
Desktop PC, Safari	✓	✓	✗	✓
Desktop PC, Chrome	✓	✓	✓	✓
Desktop PC, Firefox	✗	✗	✓	✓
Desktop PC, Opera	✗	✗	✓	✗
Desktop PC, Maxthon	✓	✓	✓	✓

Tabletti- ja mobiililaitteet (ARM), audioformaatit

Tunniste	**MP3**	**M4A**	**Ogg**	**WAV**
iPhone 5 (6.1)	✓	✓	✗	✓
iPad Mini (6.1)	✓	✓	✗	✓
iPad 3 (6.1)	✓	✓	✗	✓
iPad 2 (6.1)	✓	✓	✗	✓
iPhone 5 (6.0.2)	✓	✓	✗	✓
iPhone 4S (6.0.1)	✓	✓	✗	✓
iPhone 4 (6.0.1)	✓	✓	✗	✓
iPad Mini (6.0.1)	✓	✓	✗	✓
iPad 3 (6.0.1)	✓	✓	✗	✓
iPad 2 (6.0.1)	✓	✓	✗	✓
iPhone 5 (6.0)	✓	✓	✗	✓
iPhone 4S (5.1.1)	✓	✓	✗	✓
iPhone 3GS (5.1.1)	✓	✓	✗	✓
iPad 3 (5.1.1)	✓	✓	✗	✗
Galaxy S III (4.0.4)	✓	✓	✗	✗
Xperia T (4.0.4)	✓	✓	✗	✗
One S (4.0.3)	✓	✓	✗	✗
One X (4.0.3)	✓	✓	✗	✗
One V (4.0.3)	✓	✓	✗	✗
C8812 (4.0.3)	✓	✓	✗	✓
Xoom (4.0.4)	✓	✓	✓	✗
Galaxy Tab 10.1 (4.0.4), Android Browser	✓	✓	✗	✗
Galaxy Tab 10.1 (4.0.4), Chrome	✓	✓	✓	✓
Galaxy Tab 10.1 (4.0.4), Firefox	✓	✗	✓	✓
Galaxy Tab 10.1 (4.0.4), Opera Mobile	✓	✓	✓	✓
Galaxy Tab 10.1 (4.0.4), Dolphin	✓	✓	✗	✗
Galaxy Tab (2.3.6)	✓	✓	✗	✗
Windows Phone 8X (8)	✓	✓	✗	✗
Lumia 900 (7.5)	✓	✓	✗	✗

Kannettavat ja työpöytäkoneet (x86), videoformaatit

Tunniste	**H.264/MPEG-4**	**Ogg/Theora**	**WebM/VP8**
MacBook Air, Safari	✓	✗	✓
MacBook Air, Chrome	✓	✓	✓
MacBook Air, Firefox	✗	✓	✓
MacBook Air, Opera	✗	✓	✓
MacBook Air, Maxthon	✓	✓	✓
Desktop PC, IE	✓	✗	✗
Desktop PC, Safari	✓	✗	✗
Desktop PC, Chrome	✓	✓	✓
Desktop PC, Firefox	✗	✓	✓
Desktop PC, Opera	✗	✓	✓
Desktop PC, Maxthon	✓	✓	✓

Tabletti- ja mobiililaitteet (ARM), videoformaatit

Tunniste	**H.264/MPEG-4**	**Ogg/Theora**	**WebM/VP8**
iPhone 5 (6.1)	✓	✗	✗
iPad Mini (6.1)	✓	✗	✗
iPad 3 (6.1)	✓	✗	✗
iPad 2 (6.1)	✓	✗	✗
iPhone 5 (6.0.2)	✓	✗	✗
iPhone 4S (6.0.1)	✓	✗	✗
iPhone 4 (6.0.1)	✓	✗	✗
iPad Mini (6.0.1)	✓	✗	✗
iPad 3 (6.0.1)	✓	✗	✗
iPad 2 (6.0.1)	✓	✗	✗
iPhone 5 (6.0)	✓	✗	✗
iPhone 4S (5.1.1)	✓	✗	✗
iPhone 3GS (5.1.1)	✓	✗	✗
iPad 3 (5.1.1)	✓	✗	✗
Galaxy S III (4.0.4)	✓	✗	✓
Xperia T (4.0.4)	✓	✗	✓
One S (4.0.3)	✓	✗	✓
One X (4.0.3)	✓	✗	✓
One V (4.0.3)	✓	✗	✓
C8812 (4.0.3)	✓	✗	✓
Xoom (4.0.4)	✓	✗	✓
Galaxy Tab 10.1 (4.0.4), Android Browser	✓	✗	✓
Galaxy Tab 10.1 (4.0.4), Chrome	✓	✗	✓
Galaxy Tab 10.1 (4.0.4), Firefox	✓	✓	✓
Galaxy Tab 10.1 (4.0.4), Opera Mobile	✓	✗	✗
Galaxy Tab 10.1 (4.0.4), Dolphin	✓	✗	✓
Galaxy Tab (2.3.6)	✓	✗	✗
Windows Phone 8X (8)	✓	✗	✗
Lumia 900 (7.5)	✓	✗	✗